Grado en Ingeniería Informática

Trabajo de Final de Grado

Compatibilizar un dispositivo con un asistente de voz

Autor:
Adrián Sorribas Segura

Supervisor:
Mihai Lupoiu
Tutor académico:
Juan Pablo Aibar Ausina

Fecha de lectura: 23 de junio de 2020
Curso académico 2019/2020

Resumen

Este proyecto trata de mejorar y facilitar la vida diaria de muchos usuarios, en especial de los que presentan dificultades al realizar actividades cotidianas como son las acciones relacionadas con los ascensores.

Para ello se desarrolla un servidor web que se conecta con un asistente de voz, permitiendo realizar las tareas de una forma más sencilla y accesible para todos los usuarios.

Es por esto que el objetivo principal de este proyecto va orientado a que se pueda compatibilizar un dispositivo de un ascensor con el asistente, pudiendo así llegar a un segmento de la población mayor y realizar acciones de una forma más sencilla.

Para conseguirlo se van a utilizar tecnologías de virtualización como *Docker*, para contener el servicio web y la base de datos *PostgreSQL*; y, por otra parte, se va a utilizar el asistente de voz *Google Home*. Además, se van a aunar ambas partes usando el lenguaje de programación *Golang*.

Palabras clave

Golang, Google Home, Asistente de voz, GSR, Docker

Keywords

Golang, Google Home, Voice Assistant, GSR, Docker

Índice general

Índice de tablas

Índice de figuras

Capítulo 1

Introducción

1.1. Contexto y motivación del proyecto

En 2007 inicia su actividad empresarial una pequeña empresa tecnológica llamada Nayar Systems [27], cuya actividad se basa principalmente en el desarrollo y la innovación en el campo de las telecomunicaciones y de la elevación. Su lema es la innovación como razón de ser y está claramente reflejado en su misión, pues consiste en liderar la innovación y el conocimiento en soluciones tecnológicas.

Desde sus inicios hasta la actualidad, Nayar Systems ha logrado posicionarse como una de las principales empresas tecnológicas castellonenses. Los productos que desarrolla están estrechamente relacionados con el mundo de la elevación y entre ellos podemos destacar los siguientes:

- **72horas:** Se trata de uno de los productos estrella de Nayar Systems puesto que nace con el fin de ser líder en el cumplimiento de la normativa europea EN 81-28[1] en ascensores.
- **Net4machines:** VPN *(Virtual Private Network)* de carácter privado con el objetivo de interconectar de forma segura cualquier tipo de dispositivo a Internet desde cualquier lugar del mundo.
- **Nearkey:** Aplicación móvil basada en el funcionamiento de la tecnología Bluetooth para abrir de forma segura cualquier tipo de puerta de forma remota.
- **Advertisim:** Dispositivo conectado en tiempo real capaz de mostrar información actualizada del ascensor, así como mostrar también publicidad personalizada.
- **GSR*(GSM[2] Smart Router)*:** Dispositivo capaz de interconectar todos los elementos de un ascensor mediante una tarjeta SIM haciendo uso de la tecnología IoT *(Internet of Things)*.

[1] Normativa europea que establece que los ascensores deben incorporar un sistema de alarma de comunicación bidireccional.

[2] Global System for Mobile.

Esta empresa cuenta con tres departamentos: ingeniería, informática y administración. Además, también cuenta con un departamento propio denominado *Nayar Systems Garaje*, destinado a la innovación, creación y desarrollo de nuevos productos.

El proyecto descrito en este documento se ha desarrollado en el departamento de informática y está relacionado estrechamente con dos de los productos descritos anteriormente: Advertisim y GSR.

Debido a las grandes dificultades que padecen diariamente las personas con algún tipo de discapacidad para poder usar un ascensor o cualquier dispositivo similar, el proyecto que se presenta a continuación tiene como principal objetivo la resolución de dichos problemas con el fin de facilitar la vida cotidiana de la población. Es por esto que la motivación de este proyecto viene dada por el afán de ayudar a aquel colectivo más vulnerable que presenta dificultades al realizar actividades cotidianas.

Tomando esto como punto de partida, con este proyecto se pretende realizar la integración de un asistente de voz con un dispositivo de Nayar Systems. Además, también se debe compatibilizar con el dispositivo y posteriormente ser capaces de realizar una acción mediante el asistente de voz.

1.2. Objetivos del proyecto

El objetivo principal de este proyecto es realizar la integración y compatibilizar un asistente de voz con un dispositivo de la empresa. Compatibilizar correctamente los dispositivos es una tarea necesaria para poder realizar acciones mediante el asistente.

El objetivo principal se desglosa en los siguientes objetivos parciales:

- Investigar e identificar los diferentes asistentes de voz disponibles en el mercado.
- Escoger e integrar la API (Interfaz de programación de aplicaciones) del que más se adapta al proyecto.
- Diseñar e implementar una BBDD (Base de datos) para almacenar los usuarios y sus respectivos dispositivos.
- Diseñar e implementar un servidor web para gestionar la interacción entre los usuarios y el asistente de voz.
- Dar de alta uno de los dispositivos de Nayar Systems conectado a un ascensor.
- Ser capaz de realizar al menos una acción con el asistente de voz escogido.

El principal resultado esperado es que un usuario cualquiera sea capaz de realizar una acción mediante el asistente de voz escogido y que dicha acción resulte satisfactoria.

1.2.1. Alcances

El alcance del proyecto se focaliza en compatibilizar el asistente de voz con uno de los dispositivos de Nayar Systems con el fin de llegar a un amplio segmento de la población. Para ello se hace uso de tecnologías de vanguardia como *Docker*, para la virtualización; *Vue.js*, para gestionar el proceso de autenticación; y, *Golang*, como lenguaje principal de implementación.

En los siguientes aparados se va a ahondar en los distintos alcances clasificados por ámbitos.

Alcance funcional

En el ámbito funcional, el asistente debe ser capaz de cumplir con las siguientes características:

- Se debe poder autenticar a un usuario.
- El usuario debe ser capaz de consultar los dispositivos asociados.
- Se debe poder consultar el estado de un dispositivo.
- Se debe poder realizar una acción con el asistente de voz escogido.

Alcance organizativo

En el ámbito organizativo, este proyecto pretende llegar a un gran segmento de la población, tanto interno como externo de la empresa, siempre con el requisito de que el dispositivo a conectar pertenezca a Nayar Systems. Además, los usuarios deberán disponer de un dispositivo móvil compatible con un asistente de voz.

Alcance informático

En el ámbito tecnológico se va a alojar la ejecución tanto del servidor como de la base de datos en un contenedor *Docker*. Además, se va a gestionar la autenticación de los usuarios utilizando la tecnología JWT *(JSON Web Token)*, así como también se van a implementar diferentes roles para los distintos usuarios según su autoridad. Finalmente, la comunicación asistente-servidor-GSR se va a gestionar mediante la librería *Gobbus*[3] de Nayar Systems, la que permitirá realizar las acciones con el dispositivo.

[3]Establece un flujo constante de comunicación con el dispositivo GSR.

1.3. Estructura de la memoria

La estructura de la memoria presente en este documento va a ser segmentada en diferentes capítulos cuyo contenido se detalla a continuación.

En el capítulo 2 se muestra en profundidad la descripción del proyecto junto con una descripción de las tecnologías y el hardware utilizado.

En el capítulo 3 se detalla la metodología utilizada durante la estancia, la planificación tanto inicial como final del proyecto, los recursos derivados del desarrollo del proyecto y finalmente, el seguimiento que se ha realizado del mismo.

En el capítulo 4 se describe el análisis y el diseño del sistema donde se reflejarán tanto el diagrama de casos como la estructura de la base de datos.

En el capítulo 5 se mostrarán detalles de implementación y puesta en práctica del proyecto.

En el capítulo 6 se aúnan tanto las conclusiones obtenidas al finalizar el proyecto como las posibles futuras mejoras.

Para concluir, también se adjunta un anexo con información complementaria a la puesta en marcha del proyecto.

Capítulo 2

Descripción del proyecto

La situación inicial a partir de la cual se va a desarrollar el proyecto viene dada por el auge de los asistentes de voz en la actualidad, junto con una gran demanda de la sociedad a la capacidad de adaptación. Dicha necesidad se traslada a una idea de negocio que pretende satisfacer estos problemas.

Este proyecto puede entenderse de una forma más lúdica e interactiva siguiendo la Figura 2.1, donde se puede deducir el funcionamiento de cada uno de los dispositivos de Nayar Systems y cómo se integran juntamente con el ascensor. Esta figura también muestra cómo este proyecto se relaciona con los productos mencionados anteriormente puesto que precisa del GSR para obtener la información del ascensor y del Advertisim para mostrar la información e integración del asistente.

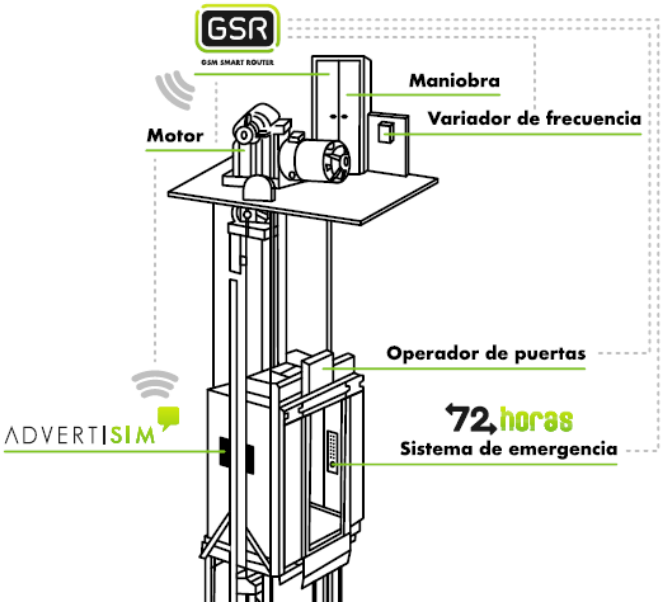

Figura 2.1: Funcionamiento de los dispositivos de Nayar Systems en un ascensor [29].

El sistema debe permitir que el usuario pueda comunicarse libremente con el asistente de voz consiguiendo, sin ningún esfuerzo físico, realizar cualquier acción permitida. Todas estas acciones van a ser especificadas expresamente al asistente de voz en el proceso de integración por lo que se podrán realizar tantas acciones como deseemos implementar. En este proyecto se ha decidido implementar la acción de apagar y encender un dispositivo GSR.

El asistente de voz se va a comunicar con el GSR mediante la librería *Gobbus* de Nayar Systems. Esta comunicación le permitirá al asistente de voz ejecutar acciones del dispositivo.

Para ello, el asistente de voz escogido ha sido el *Google Home* puesto que está presente en la gran mayoría de los dispositivos móviles. Este asistente cuenta con una serie de "intents"[1] (SYNC, QUERY, EXECUTE, DISCONNECT) [16] que son los encargados de realizar las consultas y acciones entre el servidor y el asistente. Sus funcionalidades se pueden detallar de la siguiente manera:

- **SYNC:** Se ejecuta cuando el usuario se conecta o actualiza la aplicación. Consiste en una petición que envía el asistente a nuestro servidor para obtener los dispositivos asociados al usuario.
- **QUERY:** Se ejecuta al seleccionar un dispositivo. Consiste en una petición que envía el asistente a nuestro servidor para obtener el estado del dispositivo escogido.
- **EXECUTE:** Se ejecuta al realizar una acción con el asistente. Consiste en una petición que envía el asistente a nuestro servidor junto con un comando, que debe ser tratado, para ejecutar en el GSR.
- **DISCONNECT:** Se ejecuta al desvincular una cuenta con el asistente. Consiste en una petición que envía el asistente a nuestro servidor con una señal de apagado para notificar su desconexión.

Por lo tanto el sistema funciona de la siguiente manera: El usuario ejecuta un comando de voz hablando directamente al dispositivo o pulsando sobre el mismo. Posteriormente, el asistente traduce dicho comando de voz a un objeto JSON y lo envía al asistente donde es tratado y reenvía de nuevo una acción de vuelta. Finalmente, esta acción es devuelta al dispositivo del ascensor y la ejecuta.

2.1. Tecnologías utilizadas

A continuación se muestra un listado de las tecnologías necesarias para desarrollar este proyecto así como la justificación de su uso.

[1] El asistente los envía a través del cuerpo del mensaje.

2.1.1. Asistente de voz

Una de las claves fundamentales para el desarrollo de este proyecto ha sido desde un inicio escoger un asistente de voz adecuado. Para ello se ha llevado a cabo un estudio de los diferentes asistentes de voz disponibles en el mercado así como sus alcances, su fiabilidad y su aceptación por los usuarios.

Como se puede observar en la Figura 2.2, uno de los altavoces inteligentes más usados por la población en sus hogares es *Google Home* [17], que utiliza la tecnología del asistente de voz *Google Assistant*.

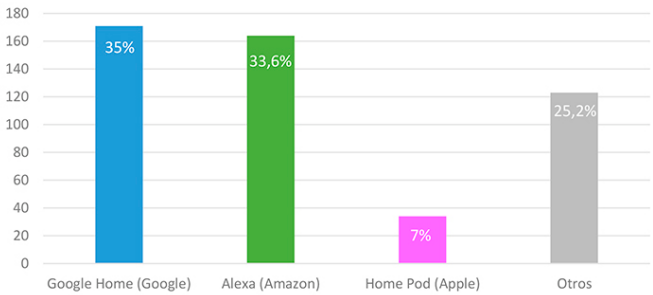

Figura 2.2: Altavoces inteligentes más usados en los hogares [20].

Este dispositivo, aparte de ser uno de los más conocidos por la población, tiene otro valor añadido pues está presente en la gran mayoría de dispositivos móviles inteligentes puesto que es compatible tanto con dispositivos *IOS* como por los que albergan el sistema *Android*.

Además de esta gran compatibilidad con los dispositivos móviles, tiene otra gran ventaja como se puede observar en la Figura 2.3, pues hace uso de uno de los asistentes de voz que mejor reconocen los patrones de voz.

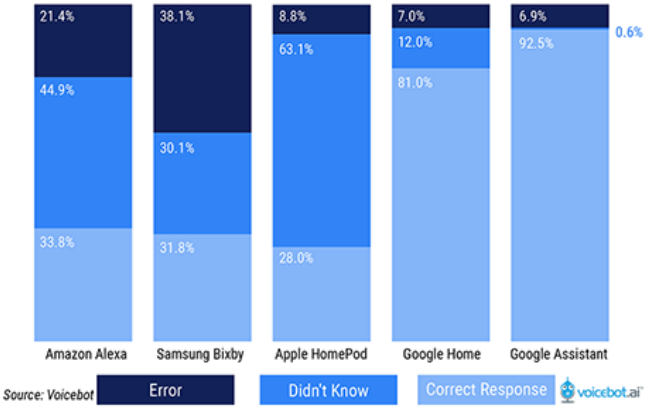

Figura 2.3: Asistentes de voz más inteligentes [7].

De este modo, también es capaz de interconectarse con *Google Assistant* mediante la configuración de *Dialogflow*[2] por lo que lo hace mucho más potente.

En definitiva, es gracias a estos indicadores por lo que se ha decidido optar por la tecnología del asistente de la marca de *Google*. Además, otro factor relevante ha sido la estrecha cercanía de los productos de Nayar Systems con la tecnología utilizada por *Google*.

2.1.2. Otras tecnologías

Golang

Golang [13] se trata de un lenguaje de programación de código libre desarrollado por *Google* y basado en el lenguaje de programación C. Además, se trata de un lenguaje compilado, concurrente, orientado a objetos y que presenta un rendimiento comparable al lenguaje C. En este proyecto es utilizado mayoritariamente para realizar la implementación del servidor y la conexión con la BBDD utilizando las librerías *Gin Gonic* [21] y *Gorm* [22] respectivamente. En definitiva, se trata del lenguaje vehicular del proyecto.

PostgreSQL

PostgreSQL [30] se trata de un sistema de gestión de bases de datos relacional orientado a objetos y de código libre. Se trata de una base de datos con una alta seguridad y que garantiza altamente la integridad de los datos. En este proyecto se utiliza para la implementación de la base de datos tanto para los usuarios del sistema como para los dispositivos asociados a dichos usuarios. Además, las transacciones que proporciona esta base de datos son de gran utilidad en la implementación de los métodos CRUD[3].

Docker

Docker [25] se trata de una tecnología de contenedores ligeros ya que contienen el *software* necesario que permiten desplegar de forma independiente aplicaciones dentro de ellos. Gracias a estos contenedores se permite que todas las dependencias de una aplicación se mantengan aisladas dentro de ese contenedor. Es por esto, que permite la automatización de la virtualización de varios sistemas operativos mediante el aislamiento de recursos del kernel. En el proyecto se utiliza dicha técnica para crear y aislar tanto la BBDD de los usuarios y dispositivos como el servidor.

[2]Herramienta de aprendizaje automático para entrenar al asistente de voz [15].
[3]Crear, Leer, Actualizar, Borrar [10].

JSON *(Javascript Object Notation)*

JSON [12] se trata de un formato de texto muy sencillo con el fin de intercambiar datos. Debido a su sencillez permite el intercambio de datos entre un servidor y un cliente de una forma muy eficaz y rápida. Es por esto por lo que en este proyecto se ha optado por la utilización de esta tecnología para realizar las transmisiones de datos entre el servidor y el asistente de voz así como también para la gestión de errores y excepciones.

Javascript

Javascript [34] se trata de un lenguaje de programación interpretado, orientado a objetos y mayoritariamente orientado a desarrollo de *scripts* para páginas web. En el proyecto se encarga de gestionar los datos que introduce el usuario a través del formulario de inicio de sesión y de enviarlos al servidor.

HTML *(Hypertext Marked Language)*

HTML [33] se trata de un lenguaje usado para estructurar y desplegar páginas web. Al tratarse de un lenguaje de programación de marcado, la unidad básica que lo compone se denomina "etiqueta". En el proyecto se ha utilizado para implementar la página de autenticación de nuestro servicio, es decir, el formulario de inicio de sesión.

CSS *(Cascade Style Sheet)*

CSS [32] se trata de un lenguaje de diseño gráfico para estilizar las páginas HTML. Es una herramienta muy potente para gestionar los estilos del *frontend*[4]. En este caso se ha usado con el fin de estilizar y animar la página de autenticación.

Vue.js

Vue.js [31] se trata de un marco de *Javascript* de código libre orientado al uso de interfaces de usuario y de muy fácil integración con el proyecto, con un enfoque reactivo y progresivo. En este caso se usa principalmente para estilizar y manejar el formulario de autenticación.

JWT *(JSON Web Token)*

JWT [6] se trata de un estándar abierto y basado en JSON que permite la autenticación, establecimiento de privilegios entre usuarios y la transmisión de información de forma segura

[4]Parte del sitio web que interactúa con los usuarios.

a través de objetos JSON. En el proyecto se utiliza primordialmente en la autenticación de usuarios y para establecer sus sesiones y delimitar sus privilegios.

Git-Flow

Git-Flow [4] se trata de un conjunto de extensiones del control de versiones *Git* con la finalidad de gestionar más cómodamente las ramas y controlar más fácilmente el flujo de trabajo. *Git-Flow* distribuye el flujo de trabajo entre las siguientes ramas: *master, develop, feature, release y hotfix*. En este proyecto se hace uso de esta tecnología centrándose en las ramas *feature, develop y master*. Cada vez que se desea implementar un hito del proyecto se establece una nueva rama *feature*. Una vez se ha cumplido con la tarea se procede a realizar un *Pull-Request* para que el supervisor decida si el trabajo realizado es correcto y debe promocionar a una rama superior, en este caso *develop*. Una vez todos los hitos son superados y todas las ramas *feature* se han cerrado se realiza otro *Pull-Request* en el que el supervisor revisa el código de la rama *develop* y si todo está correcto se realiza un *merge* con la rama *master* y se crea una *release*[5].

Jira

Jira [3] se trata de una herramienta que permite organizar y planificar el trabajo en el desarrollo de *software* de una forma eficiente y manejable. En este proyecto se ha utilizado para planificar el proyecto mediante una tabla *Kanban* en la que las diferentes tareas propuestas se han ido desplazando entre las diferentes columnas según su estado. En este caso, la pizarra se ha dividido en cuatro columnas: tareas pendientes, listo para el desarrollo, en curso y, finalmente, tareas listas.

Visual Studio Code

Visual Studio Code [24] se trata de un editor de código distribuido y creado por *Microsoft*. Este editor de código es altamente adaptable por lo que permite añadir diferentes módulos según las necesidades del proyecto. Además, es compatible con una amplia gama de lenguajes de programación por lo que al ser también de código libre hacen de él una herramienta muy propicia.

Gobbus

Gobbus [28] se trata de una librería desarrollada por Nayar Systems que utiliza el protocolo de interacción *Obbus* y que permite las comunicaciones con el dispositivo GSR usando la tecnología de *MSG-Pack*[6]. En este proyecto se utiliza para entablar comunicación y realizar acciones con el dispositivo GSR.

[5] Versión definitiva del proyecto.

[6] Formato de intercambio de datos, compacto y muy simple.

Bitbucket

Bitbucket [2] se trata de una plataforma que permite alojar y administrar los repositorios remotos. Además, también permite un control sobre el flujo de trabajo utilizando la tecnología *Git-Flow* y permite también una integración con *Jira* por lo que hace de él un *software* muy potente. En este proyecto se usa para alojar el repositorio del proyecto así como también para controlar el flujo de las ramas de *Git-Flow*.

Además, se requieren servidores para mantener el proyecto siempre operativo, en este caso el proyecto se aloja en un servidor de *Google*.

2.2. Hardware utilizado

El dispositivo empleado en este proyecto es un dispositivo GSR. Este dispositivo ilustrado en la Figura 2.4, se trata de un *router*, dispositivo de comunicaciones IoT, con el que permite entablar una conexión con la maniobra[7] de un ascensor y por ende realizar acciones con él.

Figura 2.4: Dispositivo GSR [26].

Dicho dispositivo consta de varias interfaces y en el caso que conviene a este proyecto se hace uso de la interfaz que controla el denominado *WI-FI Offline*[8] con tal de poder realizar una acción de apagado o encendido con él.

[7]Dispositivo que controla cada una de las acciones de un ascensor.
[8]Dispositivo WI-FI de menor consumo de datos.

Capítulo 3

Planificación del proyecto

3.1. Metodología

Para la realización y la consecución correcta de este proyecto, la metodología usada ha sido *Kanban*.

Kanban es una metodología en la que se crea un tablero de trabajo y un grupo de tareas pendientes [5]. En este caso se ha creado un tablero usando la aplicación web *Jira*, en el que se han organizado las tareas en cuatro columnas diferenciadas como se puede observar en la Figura 3.1.

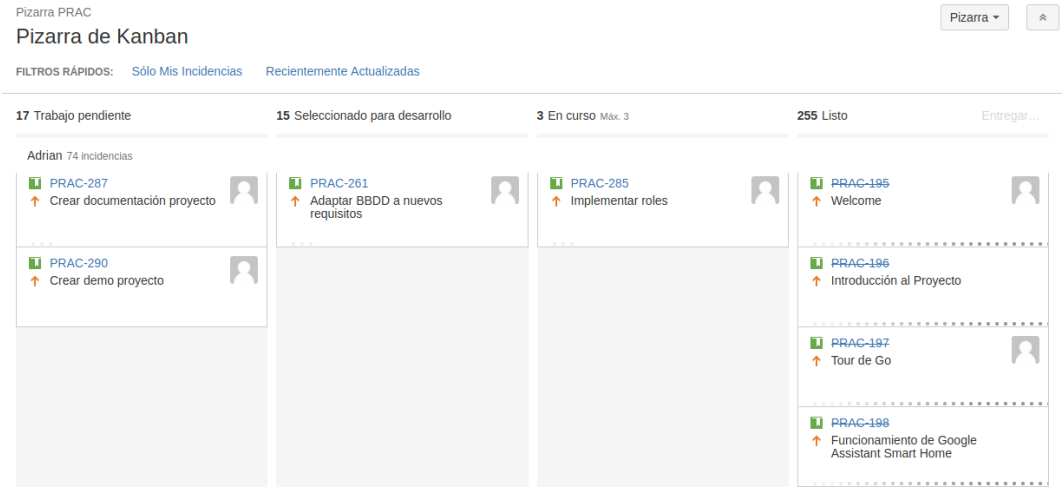

Figura 3.1: Pizarra Kanban.

Las tareas se mueven de un tablero a otro según las prioridades del desarrollador y de las actividades que se están realizando en ese momento.

Es por esto que se trata de una metodología rápida, sencilla y de fácil adaptación a los procesos de una empresa con una distribución del trabajo equitativa junto con una alta organización. Además, esta metodología es la más adecuada para este proyecto puesto que permite tener un control muy contundente de las tareas pendientes y permite mover las tareas pendientes de una columna a otra dependiendo de las necesidades de cada momento.

Las tareas que presenta esta metodología pueden ser de diferentes tipos, dependiendo de la magnitud de la tarea (si ésta se puede dividir en subtareas) o del tipo de la tarea, por ejemplo, solucionar un error surgido durante el proyecto.

Además, *Jira* también proporciona herramientas que permiten realizar gráficos para observar la evolución de las tareas del proyecto como se puede observar en la Figura 3.2, donde se muestra la evolución de las tareas que se han ido realizando durante la estancia en la empresa.

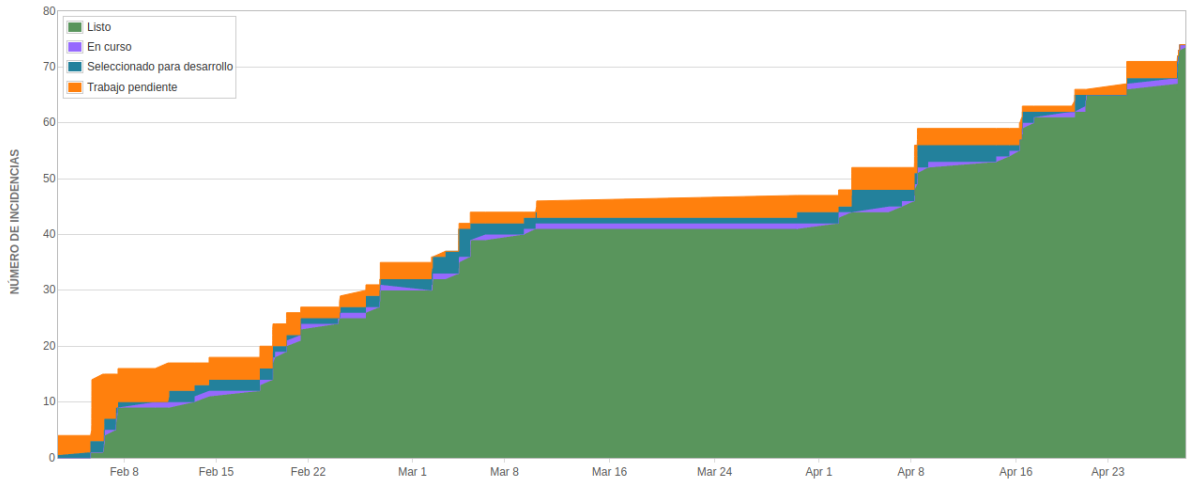

Figura 3.2: Evolución de las tareas del proyecto.

Por otra parte, la organización del código del proyecto se ha realizado a través del control de versiones *Git* usando la herramienta de control de flujo *Git-Flow* y alojando el proyecto en *Bitbucket*. Esta combinación ha permitido que a medida que se han ido superando diferentes hitos se ha invocado un *Pull-Request* que ha permitido al supervisor controlar detalladamente los cambios realizados entre distintas tareas, manteniendo así un control total entre la versión estable y la versión en desarrollo.

Como ejemplo del uso de estas tecnologías se muestra en la Figura 3.3 cómo se ha llevado a cabo el flujo entre las distintas ramas.

Figura 3.3: Cambios subidos a *Bitbucket* y evolución entre distintas ramas.

Finalmente, se han ido realizando reuniones con el supervisor semanalmente –desde el inicio de las prácticas telemáticas diariamente– para ser conocedor del estado del proyecto.

3.2. Planificación

Esta sección muestra cómo ha sido la planificación llevada a cabo durante el proyecto. Se va a dividir en dos secciones secundarias puesto que han surgido desviaciones durante la planificación inicial especificada en la propuesta técnica.

3.2.1. Planificación inicial

La planificación inicial consta de cinco puntos principales que agrupan las tareas más importantes.

El primer punto de la planificación aúna 10 horas de dedicación exclusiva a la definición del proyecto y a conocer el funcionamiento de la empresa. Dicho punto no ha sufrido modificaciones puesto que la planificación horaria ha sido seguida correctamente.

El segundo punto, cuya finalidad es gestionar la planificación del proyecto, consta de 45 horas exclusivamente dedicadas a la investigación y búsqueda de herramientas que se van a usar en el desarrollo. Dicho punto no ha sufrido muchas modificaciones, si bien es cierto que se han añadido posteriormente más tecnologías para investigar.

Por otra parte, el apartado tercero con 18 horas dedicadas al análisis y definición de requisitos del proyecto, tampoco ha sufrido grandes modificaciones ya que se han cumplido los horarios establecidos.

Por lo que respecta al punto cuarto con 173 horas asociadas a la implementación del proyecto, ha sido el punto que mayor desajuste horario ha generado. Las causas asociadas a este desajuste son las siguientes:

- Retraso en la implementación de método de autenticación.
- Retraso en la creación de la página de inicio.
- Suspensión del punto 4.3 *Implementación gRPC*[1] al no ser necesario.
- Implementación *JWT*.
- Retraso debido a la puesta en marcha de la modalidad de teletrabajo (instalación de herramientas de desarrollo e implementación en el hogar).

Finalmente, el punto número 5 con 54 horas asociadas ha presentado una reducción de horas asignadas a la puesta en marcha.

A continuación se puede observar tanto la Figura 3.4 como la Tabla 3.1 con la planificación inicial detallada.

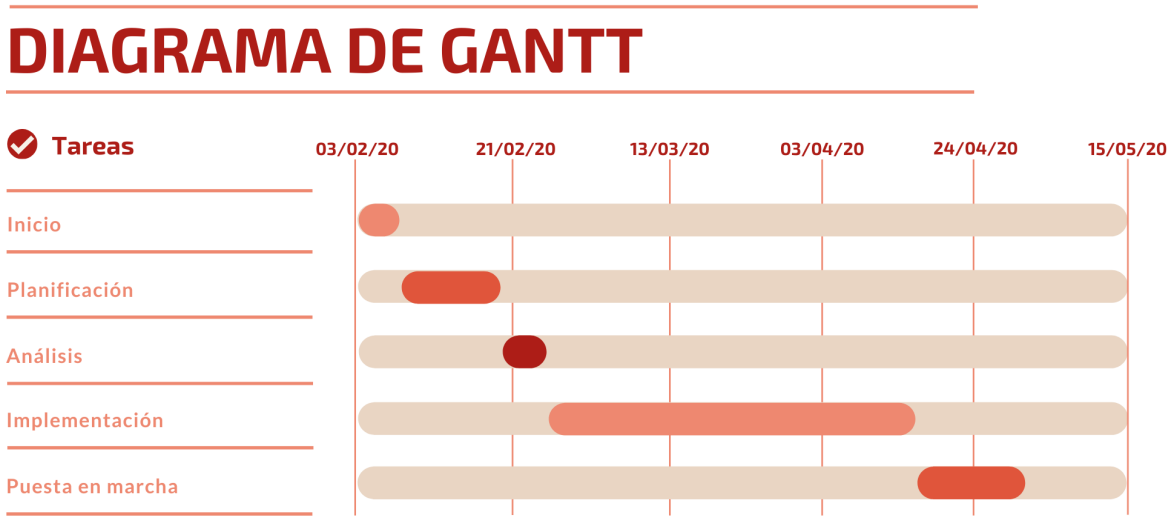

Figura 3.4: Diagrama de Gantt inicial.

[1]Protocolo RPC desarrollado por *Google* orientado a conectar microservicios.

Planificación horaria por tareas			
Nº	**Tarea**	**Tiempo (h.)**	**Dependencias**
1	**Inicio**	**10 horas**	
1.1	Definir el proyecto con el tutor y supervisor	1 hora	
1.2	Análisis del entorno	9 horas	
1.2.1	Funcionamiento de la empresa	5 horas	
1.2.2	Vista general del software de la empresa	4 horas	1.2.1
2	**Planificación**	**45 horas**	
2.1	Investigar asistentes de voz	6 horas	
2.2	Estudio de las herramientas a usar	39 horas	
2.2.1	Git Flow	3 horas	
2.2.2	Visual Studio Code	2 horas	
2.2.3	Bitbucket	2 horas	
2.2.4	Golang	12 horas	
2.2.4	JSON	4 horas	
2.2.5	Docker	4 horas	
2.2.6	gRPC	9 horas	
2.2.7	PostgreSQL	3 horas	
3	**Análisis**	**18 horas**	**2**
3.1	Crear diagrama del proyecto	9 horas	2.1
3.2	Definir requisitos tecnológicos	3 horas	2.1
3.3	Definir requisitos de uso	3 horas	
3.4	Definir requisitos de datos	3 horas	
4	**Implementación**	**173 horas**	**3**
4.1	Implementación BBDD	22 horas	
4.2	Creación servidor web	24 horas	4.1
4.3	Implementación gRPC	61 horas	4.2
4.4	Integración del asistente de voz	66 horas	4.3
5	**Puesta en marcha**	**54 horas**	**4**
5.1	Vincular un dispositivo al asistente de voz	18 horas	
5.2	Comprobar las funcionalidades del asistente	36 horas	5.1
5.2.1	Crear nuevas acciones del asistente	12 horas	
5.2.2	Probar a realizar acciones con el asistente	24 horas	

Tabla 3.1: Desglose inicial de tareas junto con la dedicación horaria y sus dependencias.

3.2.2. Planificación final

En la planificación final se han corregido todos los desfases surgidos en la planificación inicial. A continuación se explican con más detalle las modificaciones efectuadas.

El punto número uno de la planificación inicial se mantiene sin cambios.

En el apartado número dos se procede a añadir las tecnologías JWT, *Vue.js* y se descarta la investigación de *gRPC* por lo que se añaden horas a la planificación de esta tarea.

El apartado de análisis, no presenta ningún cambio horario por lo que las tareas se mantienen igual.

En la sección de implementación se descarta el apartado de implementación *gRPC* y se procede a añadir el apartado para desarrollo JWT y el de los distintos métodos que utiliza el asistente *Google Home*. Además se modifican las horas asignadas a distintas tareas.

Finalmente, en la sección quinta se procede a la reducción de horas asociadas puesto que se ha efectuado dicho punto en un menor tiempo del establecido y se desglosa una tarea en varias subtareas con el fin de establecer una mayor precisión en la planificación.

A continuación se puede observar tanto la Figura 3.5 como la Tabla 3.2 con la nueva planificación final detallada.

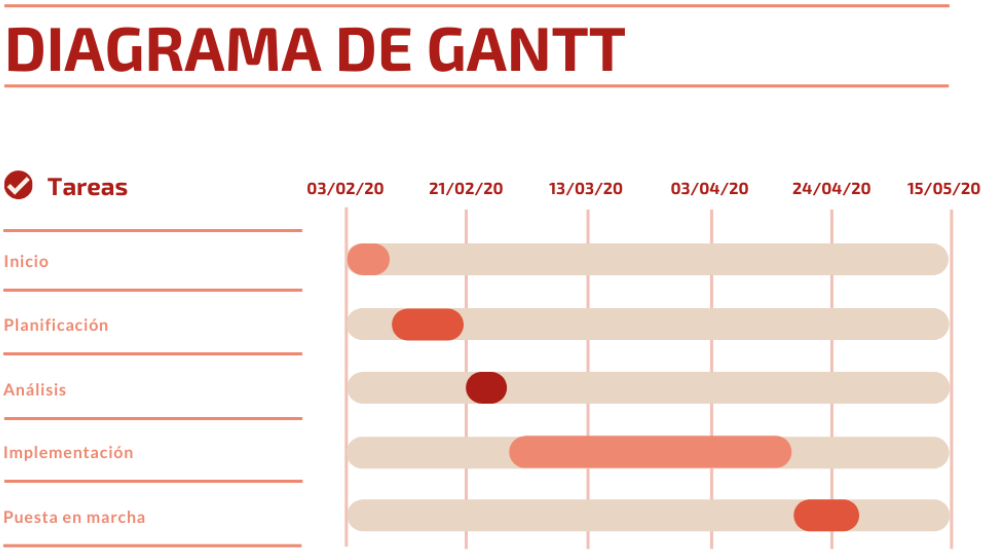

Figura 3.5: Diagrama de Gantt final.

Planificación horaria por tareas			
Nº	**Tarea**	**Tiempo (h.)**	**Dependencias**
1	**Inicio**	**10 horas**	
1.1	Definir el proyecto con el tutor y supervisor	1 hora	
1.2	Análisis del entorno	9 horas	
1.2.1	Funcionamiento de la empresa	5 horas	
1.2.2	Vista general del software de la empresa	4 horas	1.2.1
2	**Planificación**	**47 horas**	
2.1	Investigar asistentes de voz	6 horas	
2.2	Estudio de las herramientas a usar	41 horas	
2.2.1	Git Flow	3 horas	
2.2.2	Visual Studio Code	2 horas	
2.2.3	Bitbucket	2 horas	
2.2.4	Golang	12 horas	
2.2.4	JSON	4 horas	
2.2.5	Docker	4 horas	
2.2.6	JWT	8 horas	
2.2.7	Vue.js	3 horas	
2.2.8	PostgreSQL	3 horas	
3	**Análisis**	**18 horas**	**2**
3.1	Crear diagrama del proyecto	9 horas	2.1
3.2	Definir requisitos tecnológicos	3 horas	2.1
3.3	Definir requisitos de uso	3 horas	
3.4	Definir requisitos de datos	3 horas	
4	**Implementación**	**177 horas**	**3**
4.1	Implementación BBDD	30 horas	
4.2	Creación servidor web	42 horas	4.1
4.2.1	Implementar método SYNC	8 horas	
4.2.2	Implementar método QUERY	14 horas	
4.2.3	Implementar método EXECUTE	18 horas	
4.2.4	Implementar método DISCONNECT	2 horas	
4.3	Implementación JWT	45 horas	4.2
4.4	Integración del asistente de voz	60 horas	4.3
5	**Puesta en marcha**	**48 horas**	**4**
5.1	Vincular un dispositivo al asistente de voz	18 horas	
5.1.1	Realizar conexión con librería Gobbus	6 horas	
5.1.2	Gestión de los Google intents	12 horas	
5.2	Comprobar las funcionalidades del asistente	30 horas	5.1
5.2.1	Crear nuevas acciones del asistente	10 horas	
5.2.2	Probar a realizar acciones con el asistente	20 horas	

Tabla 3.2: Desglose final de tareas junto con la dedicación horaria y sus dependencias.

3.3. Estimación de recursos y costes del proyecto

Para la correcta consecución del proyecto se ha precisado de distintos recursos tanto humanos como físicos y tecnológicos.

3.3.1. Recursos humanos

En este apartado es muy importante tener en cuenta tanto la dedicación horaria que el supervisor ha desempeñado en el proyecto, como la dedicación del propio estudiante. Para realizar el estudio salarial se ha utilizado como referencia la guía salarial Hays [18] que proporciona información exacta sobre el salario medio según profesión y localidad en la que se desarrolla la acción. Para ello, con lo que respecta al salario del estudiante, se ha calculado a partir de los siguientes criterios:

- Desarrollador con menos de 2 años de experiencia.
- Zona de desarrollo profesional: Valencia.

Como objetivo se ha obtenido un salario medio bruto interanual de unos 27.000 € por lo que debe ser dividido por 253 días laborables que tiene este año para obtener el salario por hora trabajada y posteriormente dividirlo por 8 horas al tratarse de un salario a jornada completa. Como resultado de la operación se obtiene una cantidad de unos 13,33 €/h.

Por otra parte se estima que el supervisor ha empleado una hora diaria de su jornada laboral para la consecución del proyecto, por lo que al considerar que el proyecto se ha desarrollado en 50 días, hace un total de dedicación de unas 50 horas en total. Para calcular el salario medio del supervisor se ha hecho referencia a la guía anteriormente citada y se han tenido en consideración los siguientes criterios:

- Desarrollador entre 2 y 5 años de experiencia.
- Zona de desarrollo profesional: Valencia.

Usando los criterios descritos se ha obtenido un salario bruto interanual de 38000 € que realizando las ponderaciones anteriores, se obtiene un total de 18,77 €/h.

Por lo tanto el coste total de los recursos humanos se puede reflejar en la Tabla 3.3.

Al tratarse de cálculos sobre el sueldo bruto, no es preciso añadir excedentes como la seguridad social ya que ya está incluido en dichos datos.

Recurso Humano	€/h	Total horas ejecutadas	Total (€)
Sueldo estudiante	13,33	300	3999
Sueldo supervisor	18,77	50	938,5
Total			4937,5

Tabla 3.3: Desglose de gasto en recursos humanos en la estancia.

3.3.2. Recursos tecnológicos

Respecto a los recursos tecnológicos utilizados se debe diferenciar entre los recursos *hardware* y los *software*.

Los recursos derivados del *software* son nulos puesto que todas las tecnologías utilizadas en el proyecto son de código libre por lo que no computan como gastos en este caso. Entre ellas se encuentran las mencionadas en el Capítulo 2 así como también el sistema operativo utilizado, en este caso, *Arch Linux*.

Respecto a los recursos *hardware*, se desglosa en la Tabla 3.4 el uso de los mismos así como también su coste unitario.

Recurso Hardware	Cantidad	Coste unitario (€)	Total (€)
Pantalla	2	110	220
Ordenador MSI	1	200	200
Teclado	1	9,98	9,98
Ratón	1	4,99	4,99
Cable HDMI	2	4,79	9,58
Adaptador HDMI	1	4,67	4,67
Cable RJ45	1	2,70	2,70
Total			451,92

Tabla 3.4: Desglose de recursos *hardware* utilizados en la estancia y su precio total.

Además de estos costes se debe tener en consideración la vida útil de los dispositivos como su uso durante la estancia de prácticas como se puede observar en la Tabla 3.5.

Para realizar el cálculo de los costes vinculados al uso y vida útil de los dispositivos se ha realizado una estimación de la vida útil de cada uno de ellos según sus características. Además, para calcular el coste por uso se ha multiplicado el precio total de los dispositivos por el tiempo de uso, que en este caso se trata de 3 meses, dividido por la vida útil. Esto ha permitido obtener el desglose del gasto por uso durante la estancia en prácticas.

Por otra parte, en este apartado también se deben tener en cuenta los gastos derivados del uso del servidor donde está alojado el proyecto. Se ha estimado un gasto medio de 4,95 € al mes con lo que ascendería la cifra a 14,85 €.

Recurso Hardware	Total (€)	Vida útil (meses)	Coste por uso (€)
Pantalla	220	72	9,17
Ordenador MSI	200	48	12,5
Teclado	9,98	84	0,36
Ratón	4,99	36	0,42
Cable HDMI	9,58	72	0,4
Adaptador HDMI	4,67	72	0,19
Cable RJ45	2,70	72	0,11
Total	451,92		23,15

Tabla 3.5: Desglose de recursos *hardware* según su uso y vida útil.

3.3.3. Gasto total

Al total de los gastos en recursos tanto humanos como físicos se debe sumar un porcentaje que ha sido estimado en un 15 % del total en el que se encuentran gastos indirectos como son el consumo eléctrico, conexión a internet, agua, gas y alquiler del edificio.

Finalmente, en la Tabla 3.6 se encuentra desglosado el coste total del proyecto sumando todos los factores influyentes.

Recurso	Total (€)
Recursos humanos	4937,5
Recursos tecnológicos	23,15
Alquiler servidor	14,85
Recursos indirectos	810,64
Total	5786,14

Tabla 3.6: Gasto total del proyecto.

3.4. Gestión de riesgos

Para la correcta consecución del proyecto se ha realizado un análisis sobre los posibles riesgos que pueden reflejarse en una variación de la planificación del mismo. De este modo, en las tablas 3.7, 3.8 y 3.9 se muestra en detalle la evaluación de dichos riesgos así como los posibles planes de prevención y contingencia.

GR01 - Cambio de los requisitos iniciales
ID: GR01 Nombre: Cambio de los requisitos iniciales
Impacto: Variable, depende del estado en el que se encuentra el proyecto
Descripción: Un cambio tardío de los requisitos del sistema puede ocasionar una alta reestructuración del proyecto. Plan de prevención: Se debe dedicar un buen tiempo antes de empezar el proyecto en especificar unos requisitos sólidos y consistentes. Plan de contingencia: Se deben reestructurar los requisitos aprovechando al máximo el trabajo realizado y optimizando el tiempo.

Tabla 3.7: Gestión de riesgo GR01 - Cambio de los requisitos iniciales.

GR02 - Escasez de tiempo para realizar el proyecto
ID: GR02 Nombre: Escasez de tiempo para realizar el proyecto
Impacto: Bajo
Descripción: Escasez del tiempo del proyecto debido a una mala planificación. Plan de prevención: Utilizar una buena herramienta de seguimiento continuo del proyecto que permita controlar las desviaciones temporales. Plan de contingencia: Reestructurar la planificación con el fin de obtener tiempo remanente de otras actividades.

Tabla 3.8: GR02 - Escasez de tiempo para realizar el proyecto.

GR03 - Dispositivo no compatible con el asistente
ID: GR03 Nombre: Dispositivo no compatible con el asistente
Impacto: Alto
Descripción: El dispositivo proporcionado no logra conectarse con el asistente o no es reconocido por el mismo. Plan de prevención: Comprobar que la tecnología del dispositivo es compatible con el asistente o que exista un dispositivo similar compatible. Plan de contingencia: Reestructurar el proyecto para que se adapte a un dispositivo compatible certificado.

Tabla 3.9: GR03 - Dispositivo no compatible con el asistente.

3.5. Seguimiento del proyecto

El seguimiento del proyecto se ha llevado a cabo siguiendo escrupulosamente la planificación inicial y el diagrama de Gantt.

Durante la estancia se han ido realizando reuniones semanalmente con el supervisor donde se han comunicado los avances realizados hasta el momento. En algún caso se ha necesitado pivotar desde la idea inicial por lo que se ha ido adaptando la planificación a las necesidades que han surgido.

Además, tras la consecución de las tareas más importantes se han realizado *Pull-Request* con lo que el supervisor valora si los cambios realizados son correctos o si se deben introducir las modificaciones necesarias.

Tras el paso de las prácticas a modalidad telemática, las reuniones semanales con el supervisión se han convertido en diarias con tal de proporcionar una mayor cobertura en cuanto a los cambios del proyecto.

Capítulo 4

Análisis y diseño del sistema

4.1. Análisis del sistema

En esta sección se va a profundizar en el análisis tanto de los casos de uso del sistema como de los requisitos de datos.

4.1.1. Casos de uso

Los casos de uso del proyecto son de gran utilidad ya que permiten realizar un modelado del comportamiento del sistema. Para ello es necesario identificar antes que nada los actores del sistema como se muestra en la Tabla 4.1.

Actor primario	**Descripción**
Visitante	Este actor representa cualquier persona externa al sistema que es capaz de registrarse en el mismo.
Cliente	Este actor representa un cliente registrado que puede gestionar tanto sus datos personales como los de sus dispositivos y realizar acciones con el asistente.
Administrador	Este actor permite gestionar todos los usuarios del sistema así como los dispositivos asociados a ellos.
Asistente de voz	Este actor se corresponde con el asistente de voz y trata de gestionar las comunicaciones con el sistema.

Tabla 4.1: Descripción de los actores del sistema.

Por otra parte, con tal de obtener una información más detallada se muestra en la Figura 4.1 el esquema de los casos de uso del sistema implementado.

Además, en la Tabla 4.2 se muestran detalladamente los casos de uso asociados para cada usuario.

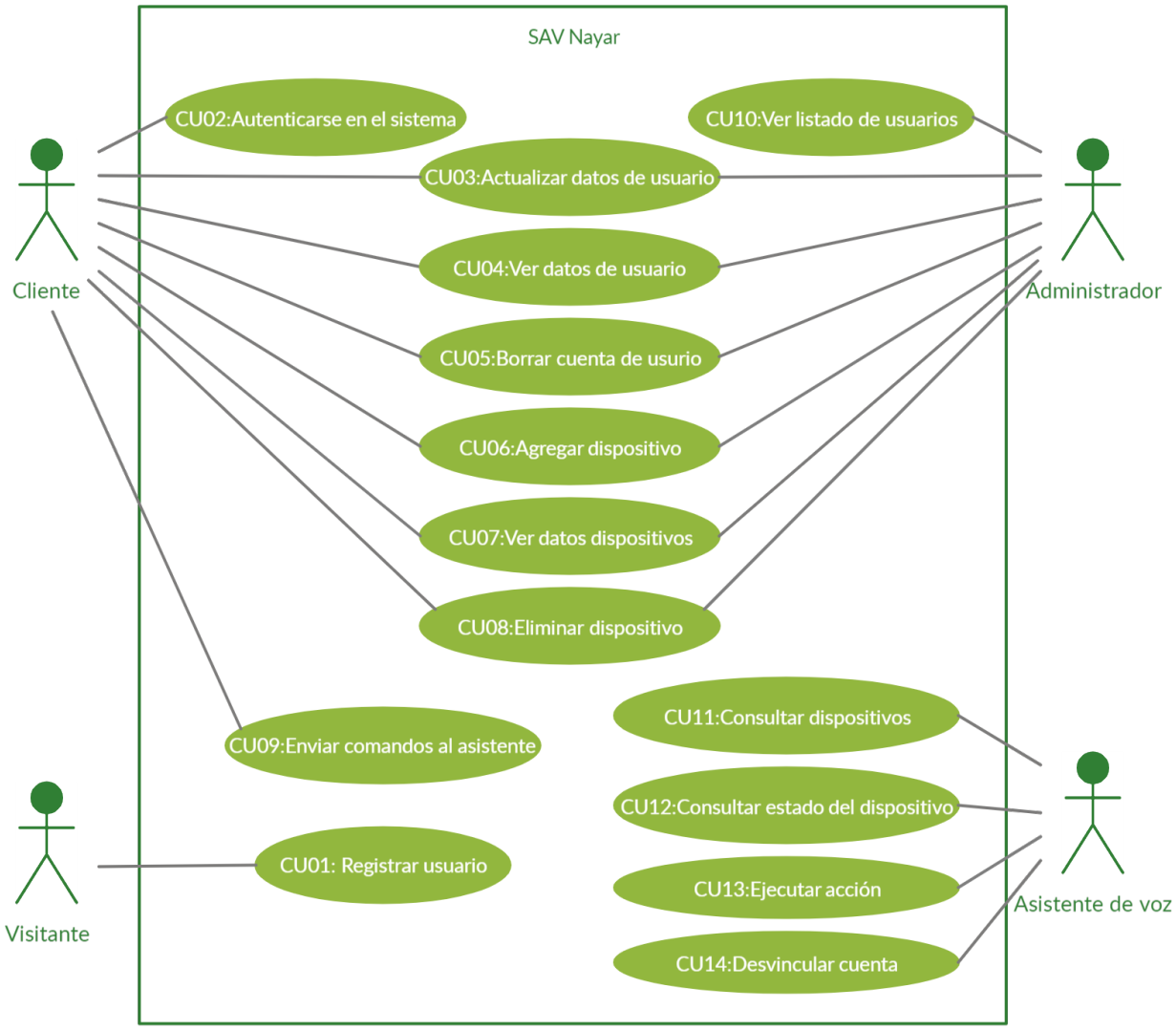

Figura 4.1: Diagrama de casos de uso.

Actor primario	**Casos de uso**
Visitante	CU01: Registrar usuario.
Cliente	CU02: Autenticarse en el sistema. CU03: Actualizar datos de usuario. CU04: Ver datos de usuario. CU05: Borrar cuenta de usuario. CU06: Agregar dispositivo. CU07: Ver datos de dispositivos. CU08: Eliminar dispositivo. CU09: Enviar comandos al asistente.
Administrador	CU03: Actualizar datos de usuario. CU04: Ver datos de usuario. CU05: Borrar cuenta de usuario. CU06: Agregar dispositivo. CU07: Ver datos de dispositivos. CU08: Eliminar dispositivo. CU10: Ver listado de usuarios.
Asistente de voz	CU11: Consultar dispositivos. CU12: Consultar estado del dispositivo. CU13: Ejecutar acción. CU14: Desvincular cuenta.

Tabla 4.2: Actores y casos de uso asociados.

Con lo que respecta a los caso de uso se desarrollan detalladamente en las tablas 4.3, 4.4, 4.5, 4.6, 4.7, 4.8, 4.9, 4.10, 4.11, 4.12, 4.13, 4.14, 4.15 y 4.16. Cabe destacar que los casos de uso CU01, CU03 - CU09 han sido implementados aunque no se ha realizado la respectiva interfaz gráfica por lo que se desarrollará como posible mejora.

CU01 - Registrar un usuario
ID: CU01 **Nombre:** Registrar un usuario **Actor:** Visitante **Fuente:** Petición por un visitante **Versión:** 1
Descripción: Permite a un usuario externo de la aplicación introducir los datos necesarios para registrarse en el portal web la primera vez que accede a la misma.
Secuencia de pasos: ▪ El usuario visitante selecciona *Registro.* ▪ El sistema muestra el formulario de entrada de datos para el registro. ▪ El visitante introduce los datos. ▪ El visitante confirma y queda registrado en el sistema.
Pre-requisitos: ▪ Un visitante solo puede registrarse si aún no está en el sistema.
Excepciones: ▪ Si alguno de los datos introducidos no es correcto, el sistema muestra el error.

Tabla 4.3: Caso de uso CU01 - Registrar un usuario.

CU02 - Autenticarse en el sistema
ID: CU02 **Nombre:** Autenticarse en el sistema **Actor:** Cliente **Fuente:** Petición por un cliente **Versión:** 1
Descripción: Permite a un cliente registrado en el sistema autenticarse y acceder a sus dispositivos.
Secuencia de pasos: ▪ El cliente entra en la aplicación. ▪ El sistema muestra el formulario de entrada de datos para el inicio de sesión. ▪ El cliente introduce los datos. ▪ El cliente confirma y los dispositivos son mostrados.
Pre-requisitos: ▪ Un cliente solo puede entrar en el sistema con registro previo.
Excepciones: ▪ Si alguno de los datos introducidos no es correcto, el sistema muestra el error.

Tabla 4.4: Caso de uso CU02 - Autenticarse en el sistema.

CU03 - Actualizar datos de usuario
ID: CU03 **Nombre:** Actualizar datos de usuario **Actor:** Cliente **Fuente:** Petición por un cliente **Versión:** 1
Descripción: Permite a un cliente actualizar sus datos personales.
Secuencia de pasos: ▪ El cliente modifica los datos que desea. ▪ El cliente envía los datos. ▪ El sistema informa si se ha realizado correctamente la operación.
Pre-requisitos: ▪ Un cliente debe estar autenticado en el sistema.

Tabla 4.5: Caso de uso CU03 - Actualizar datos de usuario.

CU04 - Ver datos de usuario
ID: CU04 **Nombre:** Ver datos de usuario **Actor:** Cliente **Fuente:** Petición por un cliente **Versión:** 1
Descripción: Permite a un cliente visualizar sus datos personales.
Secuencia de pasos: ▪ El cliente realiza la consulta. ▪ El sistema muestra los datos.
Pre-requisitos: ▪ Un cliente debe estar autenticado en el sistema.

Tabla 4.6: Caso de uso CU04 - Ver datos de usuario.

CU05 - Borrar cuenta de usuario
ID: CU05 **Nombre:** Borrar cuenta de usuario **Actor:** Cliente **Fuente:** Petición por un cliente **Versión:** 1
Descripción: Permite a un cliente borrar su propia cuenta.
Secuencia de pasos: ▪ El cliente realiza la petición de borrado. ▪ El sistema informa si se ha borrado correctamente.
Pre-requisitos: ▪ Un cliente debe estar autenticado en el sistema.

Tabla 4.7: Caso de uso CU05 - Borrar cuenta de usuario.

CU06 - Agregar dispositivo
ID: CU06 **Nombre:** Agregar dispositivo **Actor:** Cliente **Fuente:** Petición por un cliente **Versión:** 1
Descripción: Permite a un cliente agregar un dispositivo.
Secuencia de pasos: ▪ El cliente realiza la petición de agregado introduciendo los datos del dispositivo. ▪ El sistema informa si se ha agregado correctamente.
Pre-requisitos: ▪ Un cliente debe estar autenticado en el sistema.
Excepciones: ▪ Si el dispositivo ya existe, no se inserta y se actualiza.

Tabla 4.8: Caso de uso CU06 - Agregar dispositivo.

CU07 - Ver datos de los dispositivos
ID: CU07 **Nombre:** Ver datos de los dispositivos **Actor:** Cliente **Fuente:** Petición por un cliente **Versión:** 1
Descripción: Permite a un cliente ver los datos de los dispositivos asociados.
Secuencia de pasos: ▪ El cliente realiza la consulta. ▪ El sistema muestra los datos de los dispositivos asociados.
Pre-requisitos: ▪ Un cliente debe estar autenticado en el sistema.
Excepciones: ▪ Si el dispositivo no existe, se muestra un error.

Tabla 4.9: Caso de uso CU07 - Ver datos de los dispositivos.

CU08 - Eliminar dispositivo
ID: CU08 **Nombre:** Eliminar dispositivo **Actor:** Cliente **Fuente:** Petición por un cliente **Versión:** 1
Descripción: Permite a un cliente eliminar un dispositivo asociado.
Secuencia de pasos: ▪ El cliente realiza la petición de borrado. ▪ El sistema responde si se ha borrado correctamente.
Pre-requisitos: ▪ Un cliente debe estar autenticado en el sistema.

Tabla 4.10: Caso de uso CU08 - Eliminar dispositivo.

CU09 - Enviar comandos al asistente
ID: CU09 **Nombre:** Enviar comandos al asistente **Actor:** Cliente **Fuente:** Petición por un cliente **Versión:** 1
Descripción: Permite a un cliente enviar comandos para la ejecución de acciones.
Secuencia de pasos: ▪ El cliente pulsa sobre un dispositivo. ▪ El cliente realiza una acción sobre el dispositivo. ▪ El asistente devuelve si se ha ejecutado correctamente.
Pre-requisitos: ▪ Un cliente debe estar autenticado en el sistema.

Tabla 4.11: Caso de uso CU09 - Enviar comandos al asistente.

CU10 - Ver listado de usuarios
ID: CU10 **Nombre:** Ver listado de usuarios **Actor:** Administrador **Fuente:** Petición por un administrador **Versión:** 1
Descripción: Permite a un administrador obtener el listado y datos de todos los clientes del sistema.
Secuencia de pasos: ▪ El administrador realiza la petición. ▪ El sistema muestra el listado de usuarios.
Pre-requisitos: ▪ Un administrador debe estar autenticado en el sistema con el rol de administrador.

Tabla 4.12: Caso de uso CU10 - Ver listado de usuarios.

CU11 - Consultar dispositivos
ID: CU11 **Nombre:** Consultar dispositivos **Actor:** Asistente de voz **Fuente:** Petición por el asistente de voz **Versión:** 1
Descripción: Permite al asistente conocer los dispositivos de un cliente.
Secuencia de pasos: ▪ El asistente realiza una petición SYNC. ▪ El sistema responde enviando los dispositivos asociados al cliente. ▪ El asistente muestra los dispositivos.
Pre-requisitos: ▪ Un cliente debe estar autenticado.

Tabla 4.13: Caso de uso CU11 - Consultar dispositivos.

CU12 - Consultar estado del dispositivo
ID: CU12 **Nombre:** Consultar estado del dispositivo **Actor:** Asistente de voz **Fuente:** Petición por el asistente de voz **Versión:** 1
Descripción: Permite al asistente conocer el estado de un dispositivo.
Secuencia de pasos: ▪ El asistente realiza una petición QUERY. ▪ El sistema responde enviando el estado del dispositivo seleccionado. ▪ El asistente muestra el estado.
Pre-requisitos: ▪ Un cliente debe estar autenticado.

Tabla 4.14: Caso de uso CU12 - Consultar estado del dispositivo.

CU13 - Ejecutar acción
ID: CU13 **Nombre:** Ejecutar acción **Actor:** Asistente de voz **Fuente:** Petición por el asistente de voz **Versión:** 1
Descripción: Permite al asistente ejecutar una acción sobre un dispositivo.
Secuencia de pasos: ▪ El asistente realiza una petición EXECUTE. ▪ El sistema responde enviando el estado del dispositivo seleccionado y la acción. ▪ El asistente ejecuta la acción y cambia el estado.
Pre-requisitos: ▪ Un cliente debe estar autenticado.

Tabla 4.15: Caso de uso CU13 - Ejecutar acción.

CU14 - Desvincular cuenta
ID: CU14 **Nombre:** Desvincular cuenta **Actor:** Asistente de voz **Fuente:** Petición por el asistente de voz **Versión:** 1
Descripción: Permite al asistente desvincular una cuenta.
Secuencia de pasos: ▪ El asistente realiza una petición DISCONNECT. ▪ El sistema responde enviando un objeto JSON vacío. ▪ El asistente desvincula la cuenta.
Pre-requisitos: ▪ Un cliente debe estar autenticado.

Tabla 4.16: Caso de uso CU14 - Desvincular cuenta.

Por otra parte, cabe especificar que los casos CU03 - CU09 también han sido implementados por parte del actor administrador y realizan las mismas acciones que el actor cliente por lo que se ha prescindido ahondar en el detalle de los casos de uso para el rol administrador.

Finalmente, el desarrollo de esta sección se ha inspirado en el material proporcionado por la asignatura *Fonaments d'enginyería del programari* [8].

4.1.2. Requisitos de datos

En esta sección se van a mostrar los requisitos de datos necesarios de los que va a hacer uso el sistema. Para ello se pueden observar de una manera más detallada las tablas 4.17, 4.18, 4.19, 4.20, 4.21, 4.22 y 4.23 donde se muestra el flujo de datos para las diferentes acciones que permite el sistema.

RD01 - Registrar/actualizar usuario
ID: RD01 Nombre: Registrar/actualizar usuario Fuente: CU01, CU03
Tipo: Datos de entrada/salida
Descripción: Datos sobre el usuario que se va a insertar/modificar. Datos: name, lastname, email, password, role

Tabla 4.17: Requisito de datos RD01 - Registrar/actualizar usuario.

RD02 - Autenticar usuario
ID: RD02 Nombre: Autenticar usuario Fuente: CU02
Tipo: Datos de entrada
Descripción: Datos sobre el usuario que se va a autenticar. Datos: email, password

Tabla 4.18: Requisito de datos RD02 - Autenticar usuario.

RD03 - Ver datos de usuario
ID: RD03 Nombre: Ver datos de usuario Fuente: CU04, CU10
Tipo: Datos de salida
Descripción: Datos sobre el usuario. Datos: name, lastname, email, role

Tabla 4.19: Requisito de datos RD03 - Ver datos de usuario.

RD04 - Agregar dispositivo
ID: RD04 Nombre: Agregar dispositivo Fuente: CU06
Tipo: Datos de entrada
Descripción: Datos del dispositivo que se va a insertar. Datos: id, type, traits, name, nicknames, defaultnames, ip

Tabla 4.20: Requisito de datos RD04 - Agregar dispositivo.

RD05 - Obtener datos de dispositivos
ID: RD05 Nombre: Obtener datos de dispositivos Fuente: CU07
Tipo: Datos de salida
Descripción: Datos de los dispositivos asignados. Datos: id, type, traits, name, nicknames, defaultnames, ip

Tabla 4.21: Requisito de datos RD05 - Obtener datos de dispositivos.

RD06 - Consultar datos del asistente
ID: RD06 Nombre: Recibir datos del asistente Fuente: CU11, CU12
Tipo: Datos de entrada
Descripción: Datos de los dispositivos vinculados al asistente. Datos: id

Tabla 4.22: Requisito de datos RD06 - Consultar datos del asistente.

RD07 - Enviar datos al asistente
ID: RD07 Nombre: Enviar datos al asistente Fuente: CU09, CU13
Tipo: Datos de salida
Descripción: Datos del dispositivo sobre el que se realiza una acción. Datos: id, command

Tabla 4.23: Requisito de datos RD07 - Enviar datos al asistente.

4.1.3. Diagrama de actividades

Con el fin de entender mejor el funcionamiento de acceso al sistema de la aplicación, en esta sección se pretende incluir un diagrama de actividades para explicar la interrelación de las distintas actividades. Como se muestra en la Figura 4.2, se pretende mostrar el diagrama de actividades del proceso de autenticación donde una vez entramos en la aplicación de *Google Home* se redirige al *handler auth* donde se recuperan los datos de la cuenta de *Google*. Posteriormente se redirige al *handler login* donde se muestra un formulario de inicio de sesión. Si no se cumplen las credenciales, el formulario muestra un error de validación. En cambio, si las credenciales son correctas, se asocia un JWT y se redirige al asistente donde ya se muestran los dispositivos vinculados.

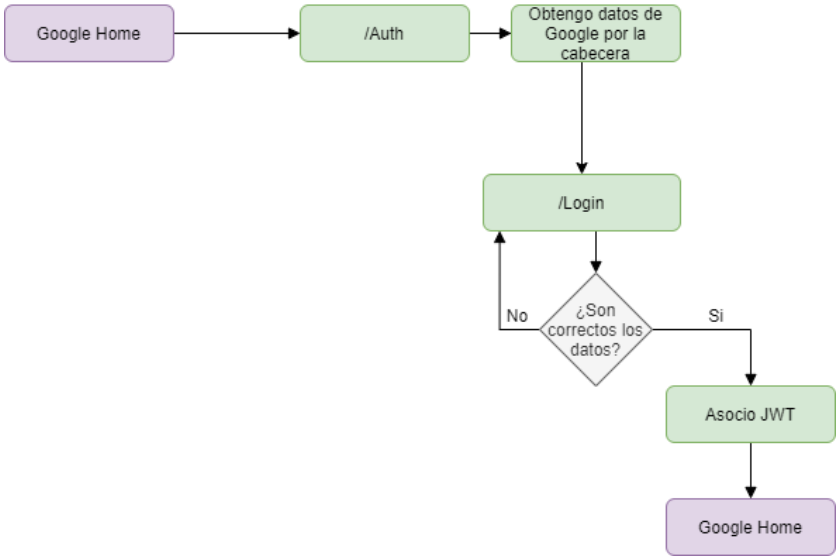

Figura 4.2: Diagrama de actividades del proceso de autenticación.

Por otra parte, también se muestran los diseños de los diagramas de actividades empleados para realizar la conexión entre el servicio web y el asistente.

Llamada SYNC

La funcionalidad de la llamada SYNC se detalla en la Figura 4.3 que funciona de la siguiente manera:

- Al principio se muestra la lista de aplicaciones compatibles.
- Una vez seleccionada la aplicación del proyecto se muestra la interfaz de autenticación.
- Se realiza una autenticación OAuth indirecta donde se genera un JWT si las credenciales son correctas.
- Se redirige a la página principal del asistente.
- Se realiza una consulta a la BBDD de la aplicación.
- Recibe la información de los dispositivos asociados y crea el *HomeGraph*[1].

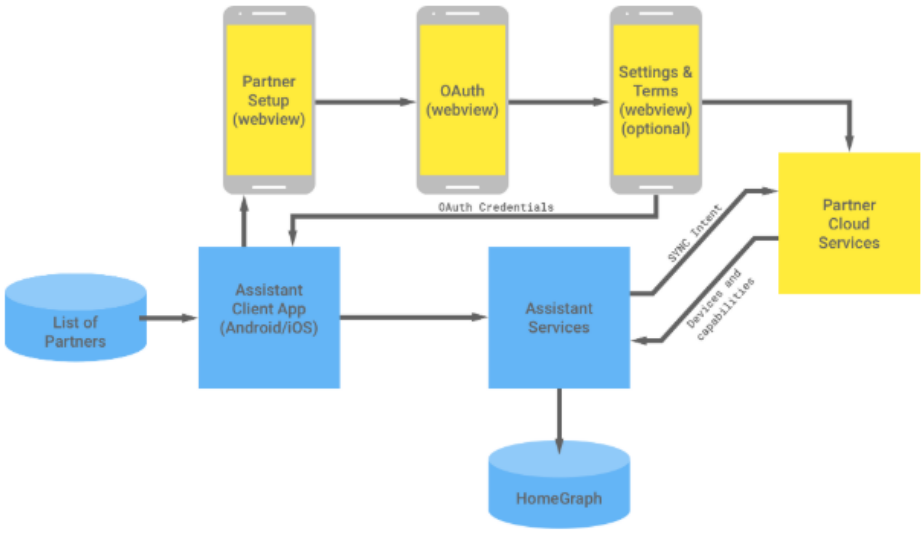

Figura 4.3: Diagrama de actividades de la llamada SYNC [16].

Llamada QUERY

Por otra parte, la funcionalidad de la llamada QUERY se detalla en la Figura 4.4 que funciona de la siguiente manera:

- Desde la pantalla principal del asistente se realiza una llamada QUERY al sistema sobre un dispositivo seleccionado.

[1]Esquema del hogar que crea el asistente a partir de los dispositivos.

- El sistema consulta el estado del dispositivo usando la librería *Gobbus* y devuelve el estado.
- El asistente actualiza el estado del dispositivo en la interfaz gráfica.

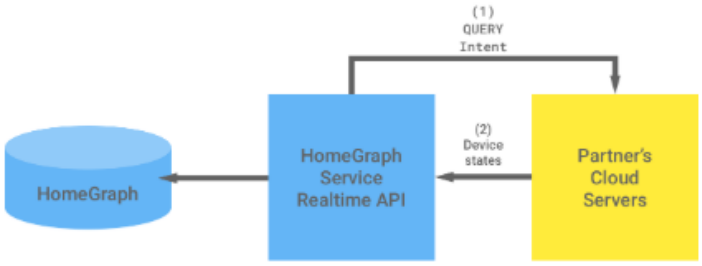

Figura 4.4: Diagrama de actividades de la llamada QUERY [16].

Llamada EXECUTE

Finalmente, la funcionalidad de la llamada EXECUTE se detalla en la Figura 4.5 que funciona de la siguiente manera:

- Desde el asistente se ejecuta un comando de voz.
- El sistema del asistente realiza reconocimiento de voz y envía los comandos obtenidos al *Google Home*.
- *Google Home* envía los comandos obtenidos a un dispositivo concreto del sistema.
- El sistema mediante la librería *Gobbus* gestiona dicho comando y ejecuta una acción sobre el dispositivo.
- Una vez ejecutado el comando se devuelve un status al *Google Home*.

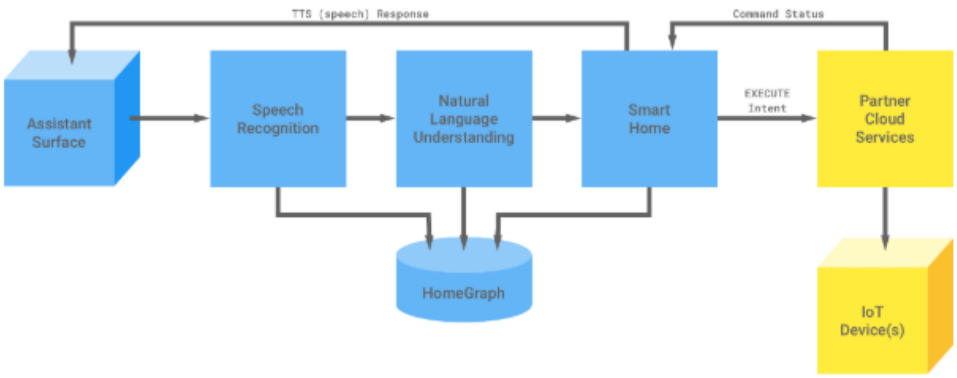

Figura 4.5: Diagrama de actividades de la llamada EXECUTE [16].

4.2. Diseño de la arquitectura del sistema

En esta sección se van a especificar funcionalidades de la base de datos así como su diseño lógico, físico y conceptual.

Como se puede observar en la Figura 4.6, la base de datos del sistema consta de dos tablas bien diferenciadas: *User* y *Device*.

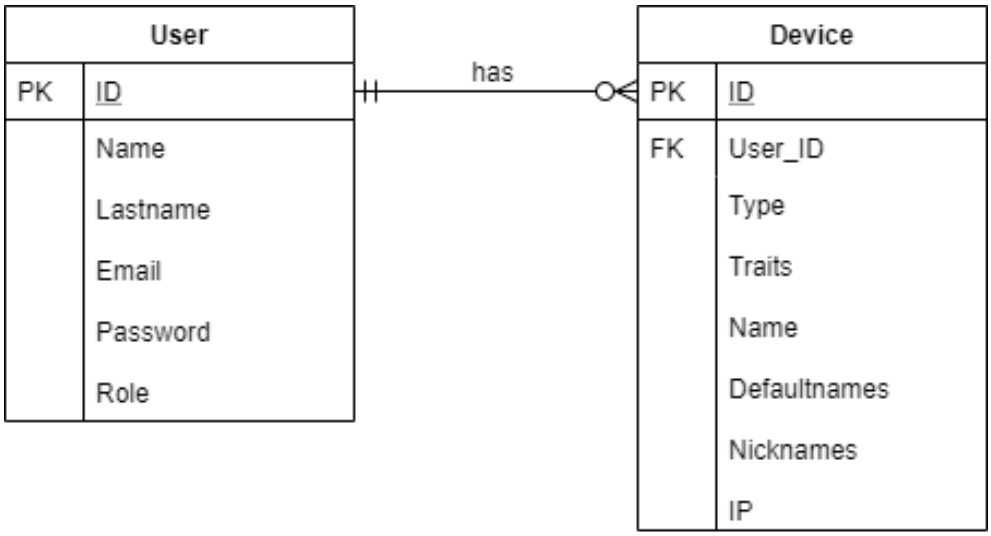

Figura 4.6: Modelo relacional de la base de datos del sistema.

Por una parte, la tabla *User* representa toda la información de un cliente que se ha registrado en el sistema.

Por otra parte, la tabla *Device* representa cada uno de los dispositivos que puede tener un cliente asociado.

4.2.1. Diseño lógico de la base de datos

En este apartado se va a mostrar el diseño lógico correspondiente a la BBDD junto con las restricciones correspondientes.

Al tratarse de una base de datos de pequeño tamaño se especifica el diseño lógico de las siguientes tablas:

User(ID, Name, Lastname, Email, Password, Role)

Device(ID, UserID, Type, Traits, Name, Defaultnames, Nicknames, IP)

Restricciones *User*: { **Borrado**: Restringir, **Modificar**: Propagar, **Nulo**: No }
Restricciones *Device*: { **Borrado**: Restringir, **Modificar**: Propagar, **Nulo**: No }

4.2.2. Diseño físico de la base de datos

En este apartado se especifica el diseño correspondiente al diseño lógico del apartado anterior.

```
CREATE TABLE user(
        ID serial PRIMARY KEY,
        name VARCHAR (80)  NOT NULL,
        first_name VARCHAR (80) NOT NULL,
        email VARCHAR (80) UNIQUE NOT NULL,
        password BYTEA NOT NULL,
        role VARCHAR (80)  NOT NULL
);
```

```
CREATE TABLE device(
        ID VARCHAR (80) PRIMARY KEY,
        userID INTEGER REFERENCES user(ID) ON UPDATE CASCADE, ON DELETE
            RESTRICT,
        type VARCHAR (100)  NOT NULL,
        traits VARCHAR (100)[] NOT NULL,
        name VARCHAR (100) UNIQUE NOT NULL,
        defaultnames VARCHAR (100)[] NOT NULL,
        nicknames VARCHAR (100)[] NOT NULL,
        IP VARCHAR (80) NOT NULL,
);
```

Finalmente, cabe destacar que este apartado se ha llevado a cabo gracias al uso del libro *Bases de datos* [23].

4.3. Diseño de la interfaz

El diseño de la interfaz del sistema de este proyecto se ha focalizado exclusivamente en el diseño de la página de autenticación ya que para el resto de actividades se usa directamente la API del asistente de voz. Como se muestra en la Figura A.1 del Anexo A, se ha decidido utilizar como estilo de interacción de la interfaz un formulario.

Para ello se han tenido en cuenta criterios para conseguir que la interfaz creada sea universal, sencilla, prevenga errores y cree una sensación de control al usuario final.

Es por esto que el formulario está provisto de múltiples etiquetas que guían correctamente al usuario indicando exactamente las acciones que se deben realizar. Además, también está provisto de un control de errores y verificación de datos mediante el uso de *Bootsrap Vue* que indica mediante diálogos el tipo de dato que se debe introducir en el campo correspondiente.

Finalmente, también se ha tenido en cuenta la paleta de colores utilizada con el fin de no dificultar la lectura e interpretación del formulario.

Capítulo 5

Implementación y pruebas

5.1. Detalles de implementación

En esta sección se detallan todos los aspectos relacionados con la implementación del proyecto. Para ello se ha decidido dividir esta sección en las diferentes tecnologías utilizadas con tal de profundizar en cada uno de los detalles de implementación.

5.1.1. Servicio virtualizado

Para la correcta consecución y despliegue de la aplicación en la nube se ha decidido hacer uso del servicio virtualizado *Docker*. Esta tecnología permite alojar la base de datos del sistema, el servidor web y el gestor de base de datos *adminer*. Además, con el fin de realizar el despliegue es necesario el uso de un fichero *docker-compose.yml* que se encarga de procesar los parámetros de configuración necesarios para cargar los servicios y crear las imágenes de los mismos.

```
# docker-compose.yml
# Running the postgres database, adminer and web service

version: '3.1'
services:
  db:
    container_name: postgresdb
    image: postgres:12.1
    restart: always
    environment:
      POSTGRES_DB: sav
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: secret

    volumes:
      - ./postgresdb:/var/lib/postgresql/data

    ports:
```

```
      - 5432:5432

  adminer:
    image: adminer
    restart: always
    depends_on:
      - db
    ports:
      - 8080:8080

  sav:
    container_name: sav
    image: sav-testing:0.0.1
    restart: always
    depends_on:
      - db
      - adminer
    volumes:
      - ./../config/dev.json:/config.json
      - ./../config/server.key:/server.key
      - ./../config/server.pem:/server.pem
      - ./../src/static:/static

    ports:
      - 8000:8000
```

Una vez detallados los parámetros de configuración necesarios para crear las imágenes se debe crear el contenedor *Docker* mediante el uso de un fichero *Dockerfile*. Posteriormente, para finalizar el proceso de despliegue en la nube se crea un archivo binario del proyecto mediante un fichero *Makefile*.

```
#Dockerfile

FROM alpine

COPY sav_unix /sav
ENTRYPOINT [ "/sav" ]
CMD ["-configFile", "/config.json", "-serverPem", "/server.pem", "-
   serverKey", "/server.key", "-static", "/static"]
```

A continuación se detallan en profundidad los comandos utilizados en el fichero *Dockerfile*:

- **FROM:** Establece la base de la imagen a partir de la que se va a partir.
- **COPY:** Especifica los datos que se van a copiar en el contenedor.
- **ENTRYPOINT:** Establece el punto de entrada del proyecto.
- **CMD:** Especifica los parámetros o comandos asociados al punto de entrada.

Una vez creada la imagen de *Docker* se ha establecido un contenedor con los recursos necesarios para que se ejecute correctamente el proyecto. Esto permite que el proyecto se mantenga siempre en marcha y bien aislado de cualquier entorno de trabajo, por lo que mejora ampliamente la seguridad del sistema.

5.1.2. Base de datos

La base de datos utilizada en este proyecto utiliza la tecnología de *PostgreSQL* puesto que se trata de una tecnología sencilla y fácil de manejar. Además, el hecho de no tratar grandes cantidades de consultas sobre la misma y gracias al uso de transacciones hacen de esta base de datos un modelo ideal para el proyecto.

Del mismo modo que se ha especificado en el apartado anterior, la base de datos se ha virtualizado mediante el uso de *Docker* y también el gestor *adminer*[1] con el que se permite la gestión de los datos de una forma más sencilla e intuitiva [1].

Respecto a la conexión entre la base de datos y el servidor web se ha hecho uso de la librería *Gorm* que permite una gestión más eficiente y sencilla de la misma. Para ello se han creado las tablas de la base de datos especificando el tipo de dato *Gorm*.

```
1 //Users Struct of the Users table
2 type Users struct {
3  ID        int    `json:"id,omitempty" gorm:"type:serial;primary_key;
     auto_increment:true"`
4  Name      string `json:"name" gorm:"type:varchar(80);NOT NULL"`
5  Firstname string `json:"firstname" gorm:"type:varchar(80);NOT NULL"`
6  Mail      string `json:"mail" gorm:"type:varchar(80);unique;NOT NULL"`
7  Password  string `json:"password,omitempty" gorm:"type:bytea;NOT NULL
     "`
8  Role      string `json:"role,omitempty" gorm:"type:varchar(80)"`
9 }
```

```
1 //Devices Struct of the devices table
2 type Devices struct {
3  ID           string         `json:"id,omitempty" gorm:"type:varchar
     (80);primary_key"`
4  Iduser       int            `json:"iduser,omitempty" gorm:"type:
     integer;primary_key;foreignkey:id;NOT NULL"`
5  Type         string         `json:"type" gorm:"type:varchar(100);NOT
     NULL"`
6  Traits       pq.StringArray `json:"traits" gorm:"type:varchar(100)[];
     NOT NULL"`
7  Name         string         `json:"name" gorm:"type:varchar(100);NOT
     NULL"`
8  Defaultnames pq.StringArray `json:"defaultnames,omitempty" gorm:"type:
     varchar(100)[];NOT NULL"`
9  Nicknames    pq.StringArray `json:"nicknames,omitempty" gorm:"type:
     varchar(100)[];NOT NULL"`
```

[1]Permite un manejo de la base de datos mediante una interfaz de usuario.

```
10    IP            string          `json:"ip" gorm:"type:varchar(80);NOT NULL
          "`
11  }
```

El funcionamiento de la conexión con el servidor funciona de la siguiente manera:

- El servidor conecta con la base de datos utilizando los parámetros de configuración establecidos.
- Se comprueba si existe la BBDD y si no existe se crean automáticamente las tablas correspondientes.
- Los servicios CRUD que permite el servidor web se realizan en su totalidad mediante transacciones para evitar lecturas fantasma, sucias e irreparables, y asegurar siempre la atomicidad y consistencia de los datos.
- Tras realizar cualquier operación con la base de datos se devuelve un objeto JSON con un código 200 si todo funciona correctamente o un código de error si no se realiza correctamente la operación.
- Finalmente, si se detiene el servicio web también se detiene la conexión con la BBDD.

Finalmente, cabe especificar que las contraseñas de los usuarios se guardan en la base de datos mediante un encriptado *SHA256*[2] [35].

5.1.3. Servidor web

El servidor web de este proyecto se gestiona mediante la librería *Gin Gonic* que proporciona el lenguaje de programación *Golang*. Se ha decidido usar esta tecnología ya que su simplicidad y seguridad permiten gestionar muy eficazmente todas las peticiones y respuestas.

En este proyecto se han creado una serie de *handlers* que gestionan tanto las conexiones con el asistente de voz como las conexiones con la base de datos.

```
1
2   //Start HTTP Server
3   r := gin.Default()
4   r.Use(Cors())
5   r.LoadHTMLGlob(*staticFolder + "/*")
6   r.Use(favicon.New(*staticFolder + "/favicon.png"))
7   r.Static("/static", "./"+*staticFolder)
8
9   //Register Handlers
10  v1 := r.Group("")
11  {
12
```

[2]Algoritmo de Hash Seguro de 256 bits.

```
13          //google handlers
14          v1.GET("/auth", google.HandlerAuth)
15          v1.POST("/", auth.TokenAuthMiddleware(), google.HandlerBase)
16
17          //Login service
18          v1.POST("/loginForm", login.HandlerLoginForm)
19          //Login without form
20          v1.POST("/login", login.HandlerLogin)
21
22          //test if app is up
23          v1.GET("/ping", func(c *gin.Context) {
24                  c.String(http.StatusOK, "OK")
25          })
26
27          //Database handlers
28          v1.POST("/users", users.HandlerInsertUser)
29          v1.POST("/users/get", auth.TokenAuthMiddleware(), users.
               HandlerGetUser)
30          v1.GET("/users/list", auth.TokenAuthMiddleware(), users.
               HandlerGetUsersList)
31          v1.PUT("/users", auth.TokenAuthMiddleware(), users.
               HandlerUpdateUser)
32          v1.DELETE("/users", auth.TokenAuthMiddleware(), users.
               HandlerDeleteUser)
33
34          v1.POST("/devices", auth.TokenAuthMiddleware(), devices.
               HandlerInsertOrUpdateDevice)
35          v1.PUT("/devices", auth.TokenAuthMiddleware(), devices.
               HandlerInsertOrUpdateDevice)
36          v1.POST("/devices/list", auth.TokenAuthMiddleware(), devices.
               HandlerGetDevicesList)
37          v1.DELETE("/devices", auth.TokenAuthMiddleware(), devices.
               HandlerDeleteDevice)
38          }
39
40          //Run in https mode
41          r.RunTLS(config.WebService.Port, *serverPem, *serverKey)
```

Con el fin de gestionar todos los *handlers* del sistema, la librería crea un denominado *router* que es el encargado de gestionar todas las peticiones. Este *router* también carga todos los ficheros estáticos necesarios para el proyecto y a su vez también se establece un *middleware* con el único fin de garantizar la seguridad.

Asimismo, en este proyecto se ha decidido agrupar todos los *handlers* para así mejorar la gestión de los mismos.

Finalmente se han creado una serie de claves de certificado SSL[3] con el objetivo final de establecer un servicio web HTTPS para que el intercambio de información se realice de forma segura y encriptada [19] [9].

[3]Secure Sockets Layer.

5.1.4. Autenticación

En esta sección se aborda la autenticación del proyecto que se gestiona mediante el uso de JWT. En este sentido se ha conseguido reforzar la seguridad del sistema y del intercambio de datos.

Asimismo, la composición de los *tokens* viene dada de la siguiente forma:

- *Header:* Se especifica el tipo de token y el algoritmo de encriptado, en este caso *SHA256*.
- *Payload:* Se especifican los datos que se envían mediante el *token*. En este caso se envía únicamente el identificador de usuario.
- *Signature:* Se especifica el tipo de firma dependiendo del tipo de encriptado.

Una vez explicadas las partes que componen un JWT, en el proyecto se ha creado un *middleware* para cada *handler*, que restringe el uso de diferentes acciones a los usuarios del sistema. De este modo se han creado dos tipos de roles: cliente y administrador.

Por otra parte, las restricciones de los clientes vienen dadas de la siguiente manera:

- Un cliente solo puede realizar acciones sobre su propio usuario.
- Un cliente solo puede acceder a los datos de sus dispositivos.

Esto se puede conseguir gracias al identificador de usuario que lleva asociado el JWT por lo que de esta manera se puede restringir las acciones de un usuario mediante el uso del *token*.

5.1.5. Interfaz gráfica

El desarrollo de la interfaz gráfica de este proyecto se ha centrado exclusivamente en la página de inicio de sesión ya que el resto del proyecto utiliza directamente la interfaz dada por el API del asistente *Google Home*. Para ello se ha hecho uso de tecnologías como HTML, para la estructuración de la página; CSS para el diseño de la misma; y finalmente, *Bootstrap Vue* para el manejo del formulario de inicio.

Además, el envío de los datos desde el cliente al servidor se ha realizado mediante el lenguaje de programación *Javascript*, que haciendo uso de un *fetch* realiza el envío de los datos a través del cuerpo del mensaje y se encarga de realizar la redirección a la página personal del asistente en el caso de que las credenciales sean correctas.

Finalmente, dicha URL de redirección es proporcionada por *Google* al seleccionar la aplicación a la que vayamos a acceder. Junto a esta URL de redirección se debe enviar el *token*

del cliente autenticado con el fin de que el asistente mantenga la sesión activa para ese usuario durante el periodo especificado (en este proyecto se especifica un tiempo de *token* de una semana).

Más detalles sobre la interfaz gráfica tanto del asistente como de la página de inicio se pueden consultar en el Anexo A.

5.1.6. Asistente de voz

Como ya se ha especificado en apartados anteriores, el asistente de voz utilizado en este proyecto se trata de *Google Home*. En lo que incumbe a este proyecto se ha tratado de compatibilizar dicho asistente con el servidor web y posteriormente realizar una acción.

Es por esto que para explicar el proceso de compatibilizar el asistente se va a desglosar este apartado en varios subapartados.

Actions Console

Se trata de una herramienta web que trata de crear, mantener, comprobar y publicar acciones[4] de *Google*.

De este modo se puede decir que se trata de la herramienta que compatibiliza el asistente de voz con el servidor web.

Esta herramienta sirve por una parte para establecer la URL concreta con la que *Google* enviará los datos de la cuenta y mediante la que al elegir la aplicación correspondiente redirigirá al usuario al formulario de inicio de sesión junto con los datos proporcionados por *Google*.

Por otra parte, también sirve para establecer el "gancho" para el servidor web mediante el que se recibirán todas las peticiones por parte del asistente, es decir, los denominados *intents*.

Además, también permite introducir el nombre con el que se invocará la aplicación en el asistente así como también especificar el tipo de autenticación que se usará en el servidor (OAuth2 u OAuth implícito).

Finalmente, en la Figura 5.1 se muestra el aspecto que tiene la consola de acciones de *Google* así como alguna de las configuraciones empleadas en el proyecto.

[4]Interacción creada para el asistente de voz.

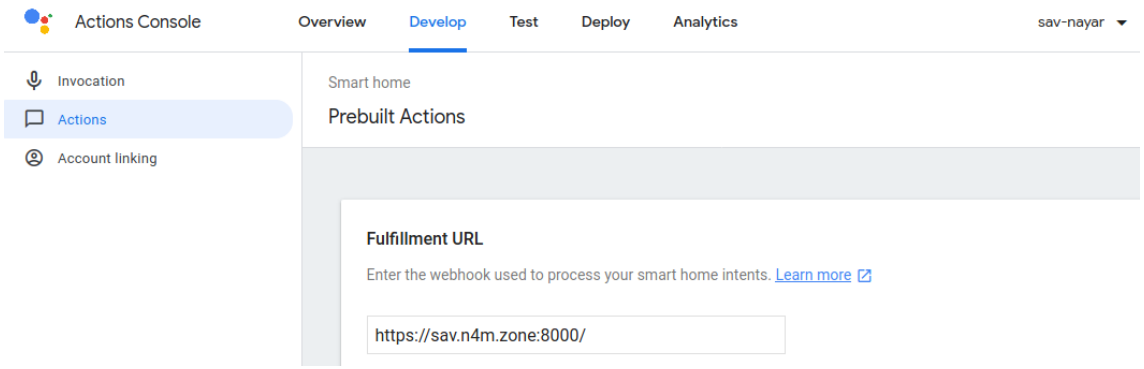

Figura 5.1: Consola de acciones de *Google Home*.

SYNC intent

La petición SYNC se invoca durante el proceso de inicio de sesión del usuario con el asistente. El asistente realiza una petición con un formato JSON como la mostrada a continuación:

```
{
    "requestId": "ff36a3cc-ec34-11e6-b1a0-64510650abcf",
    "inputs": [{
      "intent": "action.devices.SYNC"
    }]
}
```

Cuando el servidor web del sistema recibe dicha petición, la procesa y devuelve una respuesta al asistente con los datos de todos los dispositivos asociados al usuario.

Una vez devueltos los datos de los dispositivos se muestran en la página del asistente de voz. El formato de la respuesta contiene fundamentalmente dos parámetros: el identificador de la petición realizada anteriormente y un *payload* que contiene toda la información necesaria de los dispositivos.

QUERY intent

Respecto a la petición QUERY, siempre se invoca al seleccionar sobre un dispositivo disponible con el fin de conocer el estado del mismo así como conocer si el dispositivo es alcanzable o está desconectado.

Para ello, un ejemplo de petición QUERY viene dada de la siguiente forma:

```
{
    "requestId": "ff36a3cc-ec34-11e6-b1a0-64510650abcf",
    "inputs": [{
      "intent": "action.devices.QUERY",
```

```
      "payload": {
        "devices": [{
          "id": "GSR",
          }]
      }
    }]
}
```

En esta petición se especifica el identificador de la petición así como también una serie de entradas que especifican la consulta que se va a realizar y los dispositivos involucrados en la misma.

Una vez el servidor recibe el identificador del dispositivo se consulta su dirección IP en la base de datos, y posteriormente se procede a conectase con la librería *Gobbus* para que se pueda conocer el estado actual del dispositivo comprobando la conexión. Si la conexión no resulta satisfactoria se retorna un resultado de desconectado, y si lo es se retorna un estado de conectado. Además, si el dispositivo es alcanzable se consulta también el estado de la interfaz *Wifi Offline*.

En el siguiente fragmento de código se especifican los parámetros fundamentales que van a ser devueltos al asistente para que conozca el estado del dispositivo:

```
var online = false   //Information if the device is reachable {true, false}
var status = "ERROR" //Information of the status of the query device {"SUCCESS","OFFLINE","ERROR"}
var state = false    //Information if the device component is up or down {true, false}
var ip string        //ip of the device
```

EXECUTE intent

La petición EXECUTE se formaliza siempre que se pulse o se realice una acción con un dispositivo disponible. De esta forma, la petición recibida del asistente tiene el siguiente formato:

```
{
    "requestId": "ff36a3cc-ec34-11e6-b1a0-64510650abcf",
    "inputs": [{
      "intent": "action.devices.EXECUTE",
      "payload": {
        "commands": [{
          "devices": [{
            "id": "GSR"
          }, {
            "id": "GSR2"
          }],
          "execution": [{
            "command": "action.devices.commands.OnOff",
```

```
              "params": {
                "on": true
              }
            }]
          }]
        }
      }]
}
```

Consecuentemente se recibe por parte del asistente un identificador de petición y una serie de entradas donde se especifica la acción EXECUTE y un *payload* que muestra la serie de comandos a la que afecta la petición. También proporciona un apartado *execution* en el que se especifica el comando que se debe usar, que en el caso de la aplicación dicho comando se consulta a partir de la base de datos; y, una serie de parámetros permitidos por el comando y con los que se va a realizar la acción con el dispositivo GSR.

Toda esta información recibida por el asistente se trata en el servidor web y la librería *Gobbus* por lo que dependiendo de los parámetros recibidos en el campo *params* se procede a realizar una acción u otra.

Posteriormente, tras realizar una acción se devuelve mediante un objeto JSON la siguiente información:

```
var online = false   //Information if the device is reachable {true, false}
var state bool       // State of the param above (e.g. {on, off})
var status = "ERROR" //Information of the status of the query device {"SUCCESS","OFFLINE", "ERROR"}
```

Mediante estos parámetros se le indica al asistente si el comando se ha ejecutado con éxito, el estado actual del dispositivo y si sigue conectado.

DISCONNECT intent

Esta petición se invoca siempre que se desea desvincular la cuenta del asistente. Para ello se envía por parte del asistente una petición como la siguiente:

```
{
    "requestId": "ff36a3cc-ec34-11e6-b1a0-64510650abcf",
    "inputs": [{
      "intent": "action.devices.DISCONNECT",
    }]
}
```

En ella se especifica el identificador de la petición y la acción a realizar. Una vez el servidor recibe esta petición, éste le responde con un objeto JSON vacío indicando que el usuario se ha desvinculado satisfactoriamente.

5.2. Verificación y validación

Respecto a la verificación y validación de los métodos implementados en el código se ha ido comprobando su uso constantemente mediante la inyección de datos cada vez que se ha implementado un nuevo método. Es por esto que se ha ido comprobando su funcionalidad cada vez que se modifican los métodos. De este modo se ha realizado un control de errores muy exhaustivo respecto a los métodos que realizan acciones con la base de datos.

Además, el supervisor ha verificado cada uno de los métodos mediante el uso de *Pull-Request* de *Git* por lo que tras cada revisión, si los métodos están implementados correctamente y funcionan como se espera, se aceptan las modificaciones y si no se reestructura el código.

También cabe especificar que se ha realizado un control de los *logs* del proyecto utilizando la librería *Logrus* [11] de *Golang* con tal de poder clasificar el nivel de importancia de cada *log* y poder así establecer el umbral que se desee para cada ejecución del proyecto.

Finalmente, respecto a la interfaz correspondiente al inicio de sesión se ha realizado una validación de los campos del formulario mediante el uso de *Bootsrap Vue* con el fin de verificar que el tipo de datos introducido es el esperado.

5.3. Pruebas realizadas

Como ya se ha especificado en otras secciones, el dispositivo utilizado para realizar las pruebas se trata de un GSR.

Con lo que respecta al asistente de voz se ha decidido asociar el GSR como un tipo de dispositivo *OUTLET* permitiendo así realizar acciones de encendido y apagado [14]. Es por esto que se ha decidido centrar las pruebas del proyecto en realizar el encendido y apagado asociado a la interfaz que gestiona el denominado *Wifi Offline*.

Para ello, el dispositivo GSR consta de varias interfaces por lo que se ha procedido a listarlas y escoger la que mejor se adapta. Una vez encontrada y mediante el uso de la librería *Gobbus* se extrae el estado actual de la interfaz y se ejecutan las acciones correspondientes según el estado obtenido.

Con el fin de obtener un estudio más detallado de las pruebas realizadas, en el Anexo A se muestra la puesta en práctica de forma exitosa mediante el uso de la API del asistente.

Finalmente, cabe detallar que también se ha intentado utilizar otro tipo de dispositivos como el denominado *TV* y la acción *InputSelector* con tal de poder realizar acciones con el ascensor y adaptarnos a un dispositivo de este tipo. Este intento no ha resultado satisfactorio ya que al no existir un tipo de dispositivo ascensor el asistente se ve muy forzado a realizar las acciones y no funciona de forma óptima.

Capítulo 6

Conclusiones

A lo largo de esta memoria se ha descrito el proceso de compatibilizar un asistente de voz satisfactoriamente con un dispositivo conectado a un ascensor. De este modo también se han descrito los diferentes métodos que utiliza el asistente *Google Home* para comunicarse con el servidor del proyecto.

Además, la investigación en profundidad de los asistentes de voz y el desarrollo de este proyecto me ha enseñado cómo la tecnología puede llegar a ser tan necesaria en nuestro día a día.

El auge de los asistentes de voz tan solo acaba de empezar y se trata de una tecnología muy útil y muy valiosa. Este proyecto ha permitido acercarme al uso de estas tecnologías así como también a entender cómo funcionan internamente. Además, en el ámbito formativo he tenido la gran oportunidad de utilizar tecnologías como *Docker* ya que su uso va a ser muy destacado en los próximos años. Asimismo, también he tenido la oportunidad de conocer en profundidad y especializarme en el lenguaje de programación *Golang*.

Con lo que respecta al ámbito profesional, la experiencia en la empresa ha sido muy favorable puesto que me ha permitido aprender un gran abanico de tecnologías emergentes y, junto con los conocimientos aprendidos en la universidad, he mejorado con creces mis conocimientos tanto tecnológicos como profesionales ya que también he tenido la oportunidad de conocer el día a día y el funcionamiento de proyectos reales.

Finalmente, este tipo de proyectos también se puede considerar una gran oportunidad de negocio puesto que debido a las circunstancias actuales ocasionadas por el coronavirus COVID-19 se presenta una gran demanda para el uso de estas tecnologías.

6.1. Valoración personal

Mi valoración personal sobre la estancia y el proyecto es que con entusiasmo y ganas de afrontar un nuevo reto se pueden aprender muchas más tecnologías y crecer mucho más profe-

sionalmente.

El ámbito de los asistentes de voz era totalmente desconocido para mí al inicio del proyecto pero a medida que he ido investigando e indagando sobre ellos he descubierto que se trata de un mundo muy interesante y con unas tecnologías muy innovadoras así como el uso de *Machine Learning*. El uso de estas tecnologías me ha hecho crecer mucho profesionalmente puesto que he descubierto muchas tecnologías que eran totalmente desconocidas para mi.

6.2. Futuras mejoras

Como se ha especificado en el proyecto, el resultado final del mismo ha sido conseguir encender y apagar un dispositivo GSR utilizando la interfaz *Wifi Offline* mediante el uso de la librería *Obbus*.

Posteriormente se procedió a probar la realización de acciones mediante el GSR con una maniobra del ascensor situada en el edificio de Nayar Systems. Para ello se precisaba que el asistente *Google Home* tuviera un tipo de dispositivo compatible con un ascensor aunque tras realizar varios intentos con dispositivos similares que fueran capaces de realizar acciones similares a las de un ascensor (*InputSelector*) se descartó la prueba puesto que todas las acciones se realizaban de forma muy forzada.

Es por esto que por el momento el asistente *Google Home* no precisa de un dispositivo del tipo ascensor que permita realizar acciones fácilmente con este tipo de dispositivo. Sin embargo, si que es posible solicitar a *Google* que incluya este tipo de dispositivo, si bien es cierto que mucha información debe ser aportada con tal de que la propuesta sea aceptada.

Por lo tanto se considera como una posible mejora ya que si es permitida la incorporación de este tipo de dispositivo se podrá realizar cualquier tipo de acción con el ascensor.

Finalmente, también se añade como posible mejora crear una interfaz que permita la gestión de los usuarios y dispositivos de forma gráfica y no mediante línea de comandos.

Bibliografía

[1] Adminer. Database management in a single php file. https://www.adminer.org/. [Consulta: 12 de Mayo de 2020].

[2] Atlassian. A brief overview of bitbucket. https://www.atlassian.com/es/software/bitbucket/guides/getting-started/overview. [Consulta: 7 de Mayo de 2020].

[3] Atlassian. Funcionalidad de jira. https://www.atlassian.com/es/software/jira/features. [Consulta: 7 de Mayo de 2020].

[4] Atlassian. Gitflow workflow. https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow. [Consulta: 7 de Mayo de 2020].

[5] Atlassian. ¿qué es un tablero de kanban? https://www.atlassian.com/es/agile/kanban/boards. [Consulta: 7 de Mayo de 2020].

[6] Auth0. Introduction to json web tokens. https://jwt.io/introduction/. [Consulta: 7 de Mayo de 2020].

[7] Bret Kinsella. Relativo respuesta-éxito por asistentes de voz y marcas. https://voicebot.ai/2019/07/09/new-data-on-voice-assistant-seo-is-a-wake-up-call-for-brands/relative-response-success-by-voice-assistant-all-brand-product-queries-fi/. [Consulta: 22 de Mayo de 2020].

[8] Cristina Campos, Reyes Grangel, and Victòria Nebot. *Fonaments d'enginyeria del programari*. Ed. Sapientia, 2017.

[9] Chris Palmer y Matt Gaunt. Habilitación de https en tus servidores. https://developers.google.com/web/fundamentals/security/encrypt-in-transit/enable-https?hl=es. [Consulta: 12 de Mayo de 2020].

[10] Codecademy. What is crud? https://www.codecademy.com/articles/what-is-crud. [Consulta: 7 de Mayo de 2020].

[11] David Bariod. Logrus. https://github.com/sirupsen/logrus. [Consulta: 12 de Mayo de 2020].

[12] Douglas Crockford. Introducing json. https://www.json.org/json-en.html. [Consulta: 7 de Mayo de 2020].

[13] Golang. Golang documentation. https://golang.org/doc/. [Consulta: 7 de Mayo de 2020].

[14] Google. Device types and traits. https://developers.google.com/assistant/smarthome/concepts/devices-traits. [Consulta: 12 de Mayo de 2020].

[15] Google. Documentación de dialogflow. https://cloud.google.com/dialogflow/docs. [Consulta: 7 de Mayo de 2020].

[16] Google. Smart home intents. https://developers.google.com/assistant/smarthome/concepts/intents. [Consulta: 7 de Mayo de 2020].

[17] Google. Smart home overview. https://developers.google.com/assistant/smarthome/overview. [Consulta: 7 de Mayo de 2020].

[18] Hays. Guia del mercado laboral de hays 2020. https://www.hays.es/documents/63345/4314146/GUIA+DEL+MERCADO+LABORAL+DE+HAYS+2020+-+Online.pdf. [Consulta: 7 de Mayo de 2020].

[19] Heroku Dev Center . Creating a self-signed ssl certificate. https://devcenter.heroku.com/articles/ssl-certificate-self. [Consulta: 12 de Mayo de 2020].

[20] IPMARK. Los asistentes virtuales, utilizados en 4,3 millones de hogares. https://ipmark.com/los-asistentes-virtuales-utilizados-en-43-millones-de-hogares/. [Consulta: 7 de Mayo de 2020].

[21] Javier Provecho. Repositorio gin gonic. https://github.com/gin-gonic/gin. [Consulta: 7 de Mayo de 2020].

[22] Jinzhu. Repositorio gorm. https://github.com/jinzhu/gorm. [Consulta: 7 de Mayo de 2020].

[23] Mercedes Marqués Andrés. *Bases de datos*. Ed. Sapientia, 2011.

[24] Microsoft. Getting started with visual studio code. https://code.visualstudio.com/docs. [Consulta: 7 de Mayo de 2020].

[25] Mihai Lupoiu. Docker as simple as an action. https://www.youtube.com/watch?v=nMoaIyUlzIs. [Consulta: 7 de Mayo de 2020].

[26] Nayar Systems. Descripción producto gsr. https://www.nayarsystems.com/que-ofrecemos/gsm-smart-router/. [Consulta: 7 de Mayo de 2020].

[27] Nayar Systems. Historia de nayar systems. https://www.nayarsystems.com/quienes-somos/historia/. [Consulta: 7 de Mayo de 2020].

[28] Nayar Systems. Repositorio gobbus. https://github.com/nayarsystems/gobbus. [Consulta: 7 de Mayo de 2020].

[29] Nayar Systems. Catálogo nayar systems. page 9, 2020.

[30] The PostgreSQL Global Development Group. Postgresql 12.2 documentation. https://www.postgresql.org/files/documentation/pdf/12/postgresql-12-A4.pdf. [Consulta: 7 de Mayo de 2020].

[31] Vuejs. What is vue.js? https://vuejs.org/v2/guide/. [Consulta: 7 de Mayo de 2020].

[32] W3schools. Css tutorial. https://www.w3schools.com/css/. [Consulta: 7 de Mayo de 2020].

[33] W3schools. Html tutorial. https://www.w3schools.com/html/. [Consulta: 7 de Mayo de 2020].

[34] W3schools. Javascript tutorial. https://www.w3schools.com/js/. [Consulta: 7 de Mayo de 2020].

[35] Wikipedia. Secure hash algorithm. https://es.wikipedia.org/wiki/Secure_Hash_Algorithm. [Consulta: 12 de Mayo de 2020].

Anexo A

Estudio detallado de la interfaz del sistema

A.1. Formulario de inicio de sesión

En esta sección se detalla la vista correspondiente al formulario de inicio de sesión. Como se observa en la Figura A.1 se muestra el formulario vacío para comprobar la validación de los datos.

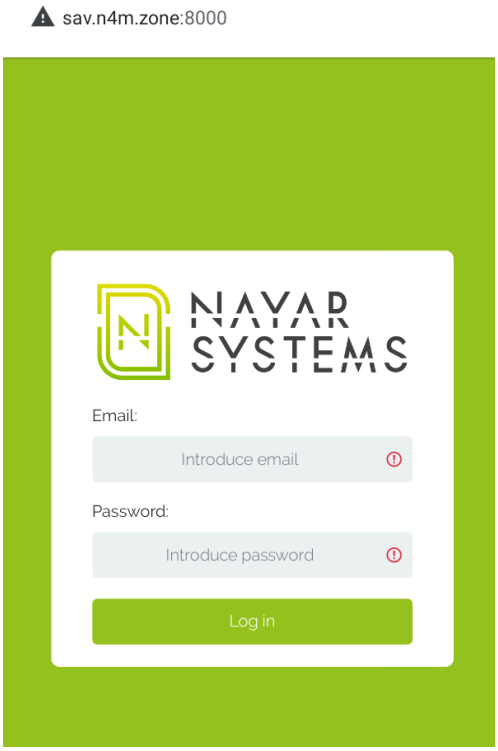

Figura A.1: Formulario de inicio de sesión.

En cambio, en la Figura A.2 se observa el formulario con los campos rellenados correctamente mostrando así la correcta validación efectuada mediante *Bootsrap Vue*.

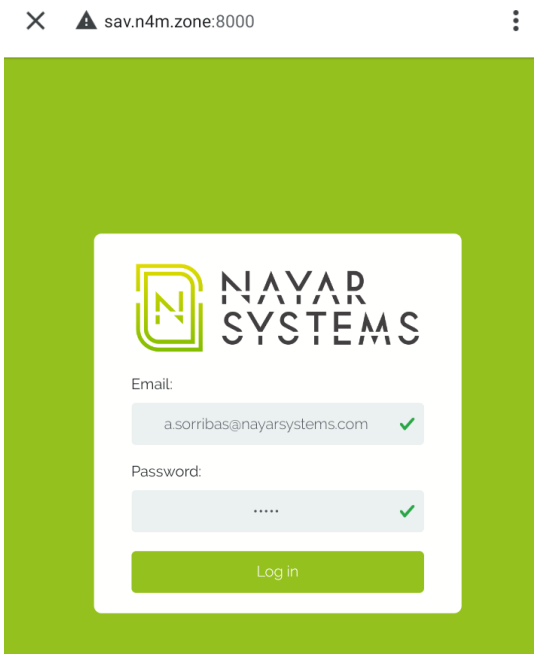

Figura A.2: Formulario de inicio de sesión con credenciales.

A.2. API del asistente *Google Home*

En esta sección se detalla el uso de la interfaz de la API del asistente *Google Home*. Es por esto que las figuras A.3, A.4, A.5, A.6 y A.7 ilustran de una forma sencilla su uso.

Al iniciar por primera vez la aplicación del asistente se muestra un listado de aplicaciones compatibles con él, como se ilustra en la Figura A.3. En este listado se muestra la aplicación *sav-nayar* ya que se trata de la creada en este proyecto. También se puede observar que no tiene un icono asociado, pues se trata de una aplicación en desarrollo.

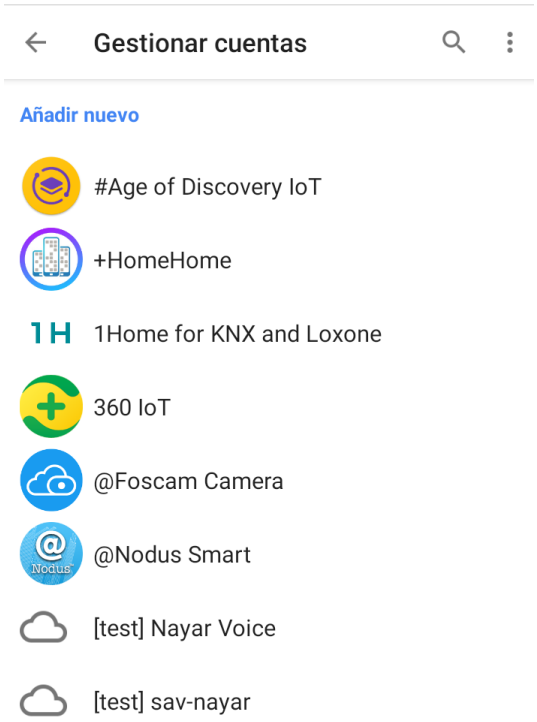

Figura A.3: Listado de aplicaciones disponibles en el asistente.

Una vez escogida la aplicación *sav-nayar* y autenticado en la misma, el asistente ejecuta una acción SYNC y como resultado se muestran los dispositivos asociados de dicho usuario como se observa en la Figura A.4.

Figura A.4: Dispositivos encontrados en la petición SYNC.

Posteriormente, en la Figura A.5 se muestra la página de inicio del usuario en el asistente de voz. En esta ocasión, el usuario ya puede empezar a realizar acciones con los dispositivos al igual que interactuar verbalmente con el asistente.

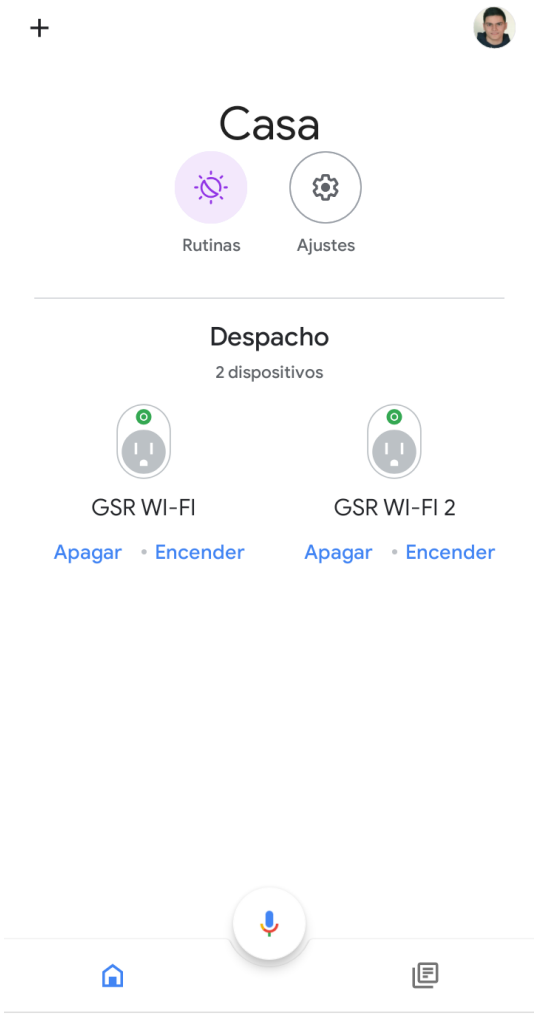

Figura A.5: Página principal del asistente.

Finalmente, al pulsar sobre un dispositivo se muestra el estado el mismo. Por ejemplo, un dispositivo con el estado de apagado se visualiza como el de la Figura A.6 mientras que un dispositivo en el estado de encendido se muestra como el de la Figura A.7.

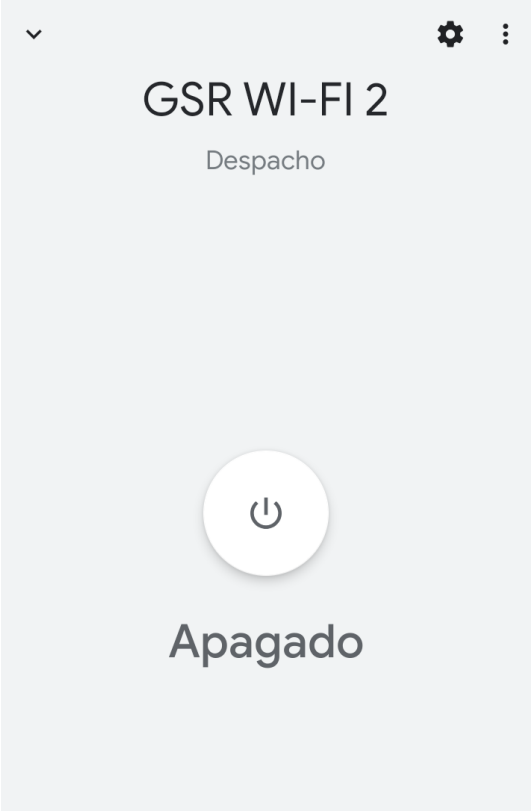

Figura A.6: Selección de un dispositivo con estado apagado.

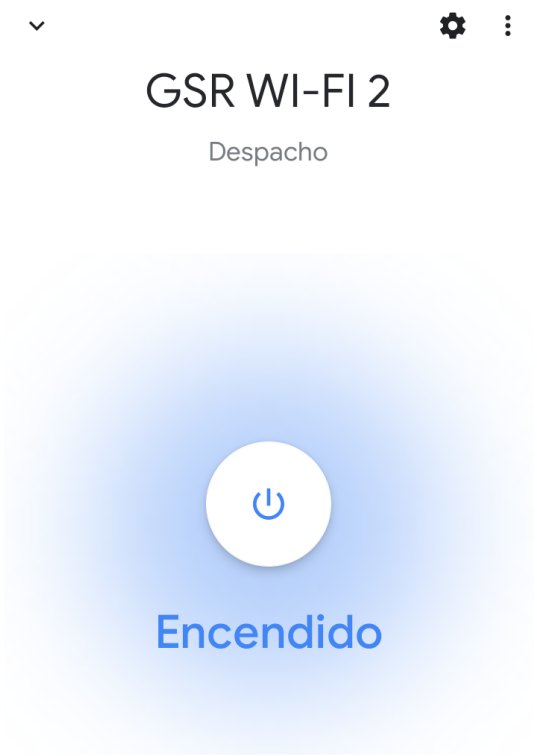

Figura A.7: Selección de un dispositivo con estado encendido.